# Chapter 8
# OSL FROM DELOCALIZED TRANSITIONS: MODELS

---

**Code 8.1: Plot of the KV-CW solution for CW-OSL in the GOT model**

```
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from scipy.special import wrightomega
def KVCW(R,lamda,j):
    c=(n0/N)*(1-R)/R
    zCWOSL=(1/c)-np.log(c)+(lamda*n0/(c*N*R))*t
    lam=wrightomega(zCWOSL)
    plt.plot(t,(N*R/((1-R)**2.0))*(1/(lam+lam**2.0)),symbs[j],
    label=labls[j]);
N,     n0 = 1e10,  1e10
t = np.linspace(0, 100, 100)
plt.subplot(1,2, 1);
symbs=['o-','+-','^-']
labls=['R=0.1','R=0.7','R=0.9']
KVCW(0.1,0.1,0)
KVCW(0.7,0.1,1)
KVCW(0.9,0.1,2)
plt.title('(a)');
plt.text(40,.6e11,'KV-CW equation');
plt.text(40,.5e11,'for CW-OSL');
plt.text(40,.4e11,'Variable R');
plt.xlabel('Time [s]');
```

```
plt.ylabel('CW-OSL [a.u.]');
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.subplot(1,2, 2);
labls=[r'$\lambda$=0.1 s$^{-1}$',r'$\lambda$=0.2 s$^{-1}$',\
r'$\lambda$=0.3 s$^{-1}$']
KVCW(0.1,.1,0)
KVCW(0.1,.2,1)
KVCW(0.1,.3,2)
plt.title('(b)');
plt.text(40,.6e10,'KV-CW equation');
plt.text(40,.5e10,'Variable power');
plt.xlabel('Time [s]');
plt.ylabel('CW-OSL [a.u.]');
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.tight_layout()
plt.show()
```

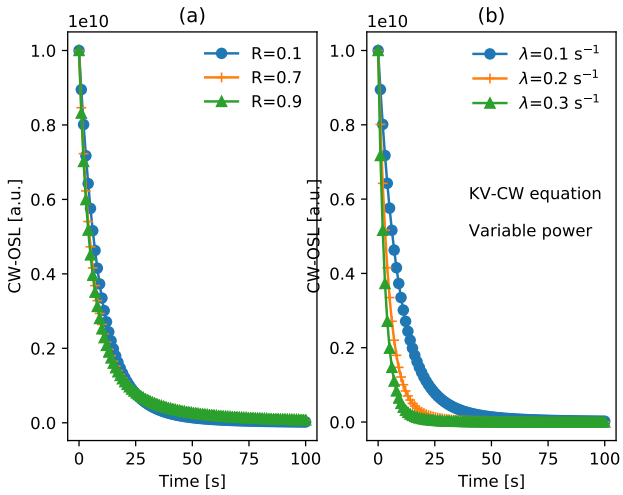


**Fig. 8.1:** (a) Plots of the KV-CW equation for three different values of the retrapping ratio $R$ =0.1, 0.7, 0.9 and for a constant excitation rate $\lambda$ =0.1 $s^{-1}$; (b) Plots for three values of the excitation constant $\lambda$ =0.1, 0.2, 0.3 $s^{-1}$ and $R$ = 0.1. The rest of the parameters are $N = 10^{10}$ $cm^{-3}$, $n_0 = 10^{10}$ $cm^{-3}$.

**Code 8.2: Plots of the MOK-CW and GOK-CW equation for CCDA**

```
#plot MOK-CW and GOK-CW equation for delocalized processes
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
def MOKCW( Nd,n):
    alpha, g =n10/(n10+Nd), 1/(N1+Nd)
    Ft=np.exp(g*Nd*lamda*t)
    CW=g*(Nd**2.0)*alpha*lamda*Ft/((Ft-alpha)**2.0)
    plt.plot(t,np.log(CW),symbs[n],\
    label=r'$\alpha$='+f'{alpha:.2f}',linewidth=2);
    return CW
def GOKCW(b,n):
    c=(n10/N1)**(b-1)
    CW=n10*c*lamda*(1+c*lamda*(b-1)*t)**(-b/(b-1))
    plt.plot(t,np.log(CW),symbs[n],\
    label=labls[n],linewidth=2);
n10, N1=1e10, 1e10
t = np.linspace(0, 130,20)
N1ds=[1e12,1e10,1e8]
symbs=['+-','^-','o-']
plt.subplot(1,2,1);
lamda=0.05
for j in range(1,4):
    MOKCW(N1ds[j-1],j-1);
plt.text(30,15,"MOK-CW");
plt.xlabel('Time [s]');
plt.ylabel('ln(CW-OSL)');
plt.title('(a)');
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.subplot(1,2,2);
lamda=0.1
Rs=[1.01,1.5,1.9]
labls=['b='+str(x) for x in Rs]
for j in range(1,4):
    GOKCW(Rs[j-1],j-1);
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.xlabel('Time [s]');
plt.text(30,11,'GOK-CW');
```

```
plt.ylabel('ln(CW-OSL)');
plt.title('(b)');
plt.tight_layout()
plt.show()
```

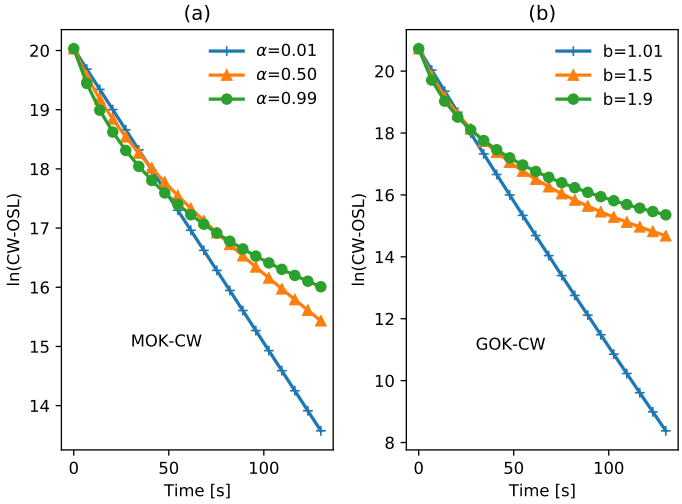


**Fig. 8.2:** Plots of (a) MOK-CW Eq.(**??**) for three values of the mixed order kinetics parameter $\alpha$ =0.01, 0.5, 0.99; (b) Plots of the GOK-CW Eq.(**??**) for $b$ =1.01, 1.5, 1.9. Note the semi-logarithmic scale.

---

**Code 8.3: Plots of the Bulur FOK equation for LM-OSL**

```
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
def BulurFOK(P,lamda,j):
    plt.plot(t,n0*lamda*t/P*np.exp(-lamda*t**2.0/(2*P)),\
    symbs[j],label=labls[j]);
n0 = 1e10
t = np.linspace(0, 100, 100)
plt.subplot(1,2, 1);
symbs=['o-','+-','^-']
```

```
labls=[r'$\lambda$=0.1 s$^{-1}$',r'$\lambda$=0.2 s$^{-1}$',\
r'$\lambda$=0.3 s$^{-1}$']
BulurFOK(100,0.1,0)
BulurFOK(100,0.2,1)
BulurFOK(100,0.3,2)
plt.title('(a)');
plt.text(60,1.9e8,'Bulur eqt.');
plt.text(60,1.7e8,'for LM-OSL');
plt.text(60,1.5e8,r'Variable $\lambda$');
plt.xlabel('Time [s]');
plt.ylabel('CW-OSL [a.u.]');
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.subplot(1,2, 2);
labls=['P=200 s','P=300 s','P=400 s']
BulurFOK(200,.3,0)
BulurFOK(300,.3,1)
BulurFOK(400,.3,2)
plt.title('(b)');
plt.text(60,1.6e8,'Variable P');
plt.xlabel('Time [s]');
plt.ylabel('CW-OSL [a.u.]');
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.tight_layout()
plt.show()
```

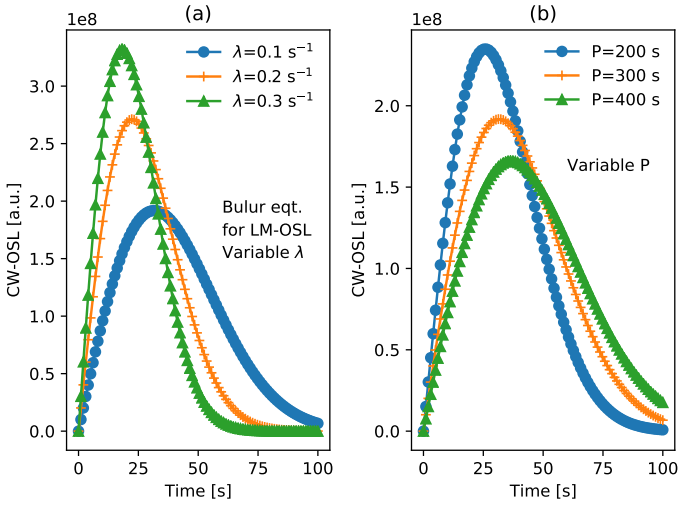

**Fig. 8.3:** Plots of the first order LM-OSL Eq.(**??**) for (a) Different values of the excitation parameter $\lambda = \sigma I = 0.1, 0.2, 0.3$ s$^{-1}$ and (b) Different values of the total excitation time $P$ and fixed $\lambda = 0.3$ s$^{-1}$.

**Code 8.4: Plots of KV-LM equation for different model parameters**

```
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from scipy.special import wrightomega
def KVLM(R,Aopt,j):
    c=(n0/N)*(1-R)/R
    zLMOSL=(1/c)-np.log(c)+Aopt*(t**2.0)/(2*P)
    lam=wrightomega(zLMOSL)
    plt.plot(t,(N*R/((1-R)**2.0))*Aopt*t/(lam+lam**2),symbs[j],
    label=labls[j]);
N,     R,  Aopt, n0 =\
1e10, 0.1, 0.3,  1e9
t = np.linspace(0, 100, 100)
P=max(t)
plt.subplot(1,2, 1);
symbs=['o-','+-','^-']
labls=['R=0.1','R=0.2','R=0.3']
KVLM(0.1,0.3,0)
KVLM(0.2,0.3,1)
KVLM(0.3,0.3,2)
plt.title('(a)');
plt.text(40,.6e11,'KV-LM equation');
plt.text(40,.5e11,'for LM-OSL');
plt.text(40,.4e11,'Variable R');
plt.xlabel('Time [s]');
plt.ylabel('LM-OSL [a.u.]');
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.subplot(1,2, 2);
labls=[r'$\lambda$=0.3 s$^{-1}$',r'$\lambda$=0.4 s$^{-1}$',\
r'$\lambda$=0.5 s$^{-1}$']
KVLM(0.1,.3,0)
KVLM(0.1,.4,1)
KVLM(0.1,.5,2)
plt.title('(b)');
plt.text(40,.6e10,'KV-LM equation');
```

```
plt.text(40,.5e10,'Variable power');
plt.xlabel('Time [s]');
plt.ylabel('LM-OSL [a.u.]');
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.tight_layout()
plt.show()
```

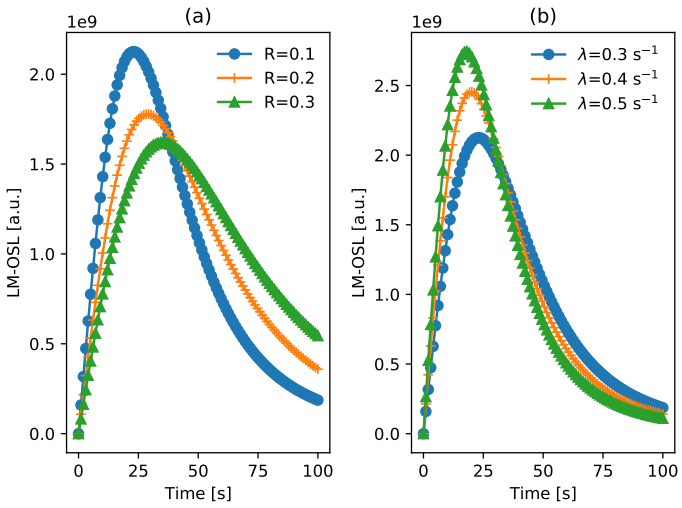


**Fig. 8.4:** (a) Plots of the KV-LM equation for different values of the retrapping ratio $R$=0.1, 0.2, 0.3 and a fixed value of $\lambda = \sigma I$ =0.3 $s^{-1}$. (b) Plots for different excitation parameters $\lambda = \sigma I$ =0.3, 0.4, 0.5 $s^{-1}$ and a fixed retrapping ratio $R = 0.1$.

---

### Code 8.5: Plots of the MOK-LM and GOK-LM equation for LM-OSL signals

```
#plot MOK-LM and GOK-LM equation for delocalized processes
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
def MOKLM( Nd,n):
```

```
    alpha, g =n10/(n10+Nd), 1/(N1+Nd)
    Ft=np.exp(g*Nd*lamda*(t**2/(2*P)))
    LM=lamda*t/(2*P)*g*(Nd**2.0)*alpha*lamda*Ft/((Ft-alpha)**2.0)
    plt.plot(t,LM,symbs[n],\
    label=r'$\alpha$='+f'{alpha:.2f}',linewidth=2);
    return LM
def GOKLM(b,n):
    c=(n10/N1)**(b-1)
    LM=n10*c*lamda*t/(2*P)*(1+c*lamda*(b-1)*\
    (t**2/(2*P)))**(-b/(b-1))
    plt.plot(t,LM,symbs[n],\
    label=labls[n],linewidth=2);
n10, N1=1e10, 1e10
t = np.linspace(0, 220,30)
P=max(t)
N1ds=[1e12,1e10,1e8]
symbs=['+-','^-','o-']
plt.subplot(1,2,1);
lamda=0.05
for j in range(1,4):
    MOKLM(N1ds[j-1],j-1);
plt.text(150,1.4e6,"MOK-LM");
plt.xlabel('Time [s]');
plt.ylabel('LM [a.u]');
plt.title('(a)');
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.subplot(1,2,2);
lamda=0.1
Rs=[1.01,1.5,1.9]
labls=['b='+str(x) for x in Rs]
for j in range(1,4):
    GOKLM(Rs[j-1],j-1)
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.xlabel('Time [s]');
plt.text(130,4e7,'GOK-LM');
plt.ylabel('LM [a.u]');
plt.title('(b)');
plt.tight_layout()
plt.show()
```

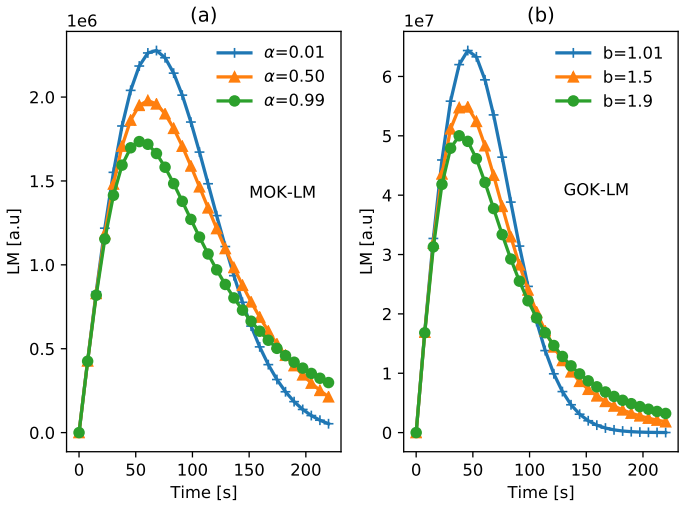


**Fig. 8.5:** Plots of (a) MOK-LM Eq.(**??**) for three values of the mixed order kinetics parameter $\alpha$ =0.01, 0.5, 0.99 and fixed excitation rate $\lambda$ =0.1 $s^{-1}$. (b) Plots of GOK-LM Eq.(**??**) for $b$ =1.01, 1.5, 1.9 and for $\lambda$ =0.1 $s^{-1}$.

---

**Code 8.6: 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()
```

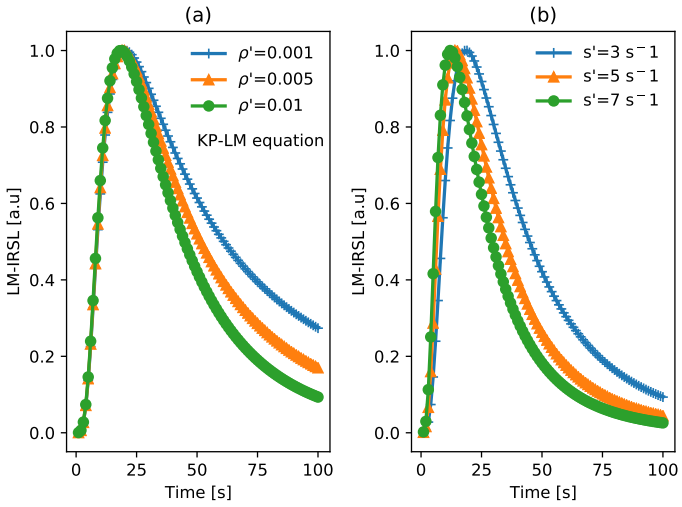


**Fig. 8.6:** (a) lot of the KP-LM Eq.(**??**) for different values of the dimensionless density $\rho'$ =0.001, 0.005, 0.01 and fixed value of $s'$ =3 $s^{-1}$.(b) Plots of the KP-LM equation for different values of the effective frequency factor $s'$ =3, 5, 7 $s^{-1}$ and $\rho' = 0.01$.

**Code 8.7: Transform CW-OSL into pseudo-LM-OSL data**

```
# Transform CW-OSL to pseudo-LMOSL for KST4 feldspar
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from scipy import optimize
from prettytable import PrettyTable
files=('OSL0s.txt','OSL100s.txt','OSL1000s.txt')
sym=('o-','+-','^-')
labs=('0 s of IR','100 s of IR','1000 s of IR')
plt.subplot(2,1, 1);
for i in range(0,3,1):
    data = np.loadtxt(files[i])
    x_data, y_data =0.1+np.array(data[:,0]),\
    0.1+np.array(data[:,1])
    plt.plot(np.log(x_data),y_data,sym[i],label=labs[i]);
plt.ylabel('CW-OSL [cts/s]');
plt.xlabel('Time [s]');
plt.title('(a)   CW-OSL: KST4 feldspar');
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.subplot(2,1,2);
for i in range(0,3,1):
    data = np.loadtxt(files[i])
    x_data, y_data=0.1+np.array(data[:,0]),\
    0.1+np.array(data[:,1])
    y_data=np.array(x_data)*np.array(y_data)
    x_data =np.log(x_data)
    plt.plot(x_data,y_data,sym[i],label=labs[i]);
plt.ylabel('I.t [a.u.]');
plt.xlabel('ln(Time)');
plt.xlim(0,9.9);
plt.ylim(0,2e7);
plt.title('(b)   Transformed CW-OSL into pseudo-LM-OSL');
leg = plt.legend(loc='right')
leg.get_frame().set_linewidth(0.0)
plt.tight_layout()
plt.show()
```

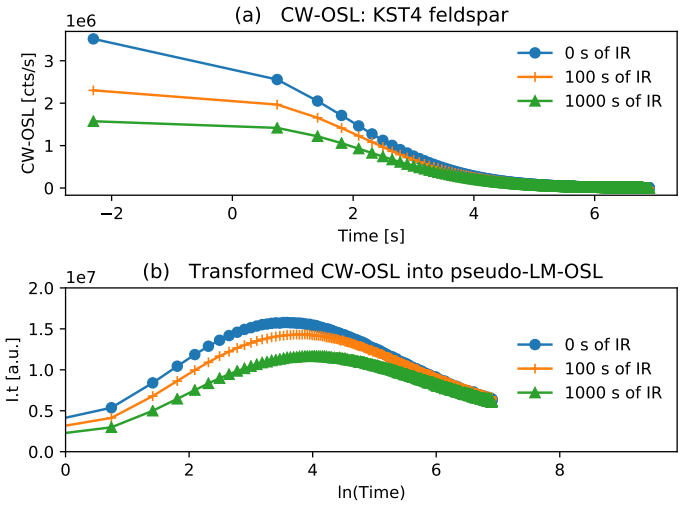


**Fig. 8.7:** Example of transforming a CW-IRSL signal into pseudo-LM-IRSL data from feldspar sample KST4. The experimental data is discussed in Pagonis et al. [**?**].

# References