Chapter 4
TL FROM QUANTUM TUNNELING PROCESSES: MODELS

Code 4.1: The nearest neighbors distribution

```
# The nearest neighbors distribution
# Fig 1 in Pagonis and Kulp paper
from scipy.spatial.distance import cdist
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure();
ax = fig.add_subplot(111, projection='3d');
ne,  nh,  side,   a=\
100, 500, 100e-9, 0.11e-9
fig = plt.figure();
ax = fig.add_subplot(121, projection='3d',xlabel='x [nm]',\
ylabel='y [nm]');
pose =side* np.random.rand(ne, 3)
posh =side* np.random.rand(nh, 3)
x,y,z=1e9*pose[:,0],1e9*pose[:,1],1e9*pose[:,2]
ax.scatter(x, y, z, c='red', marker='o');
plt.title("(a)");
x,y,z=1e9*posh[:,0],1e9*posh[:,1],1e9*posh[:,2]
ax.scatter(x, y, z, c='blue', marker='^');
ax = fig.add_subplot(122);
distances=cdist(pose, posh)
rho=nh/(side**3.0)
rhoprime=(4*np.pi*rho/3)*(a**3.0)
mindistances=1e9*np.array([min(xi) for xi in distances])
```

```
plt.hist(mindistances,12);
plt.title("(b)");
plt.tight_layout()
x=np.arange(0.0,17.0e-9,.3e-9)
rprime=(4*np.pi*rho/3)**(1/3) * x
plt.plot(x*1e9,17/0.4*(rprime**2.0)*np.exp((-rprime**3.0)),\
linewidth=2);
plt.xlabel('Nearest neighbor distance [nm]');
plt.ylabel('Distribution');
plt.tight_layout()
plt.show()
```

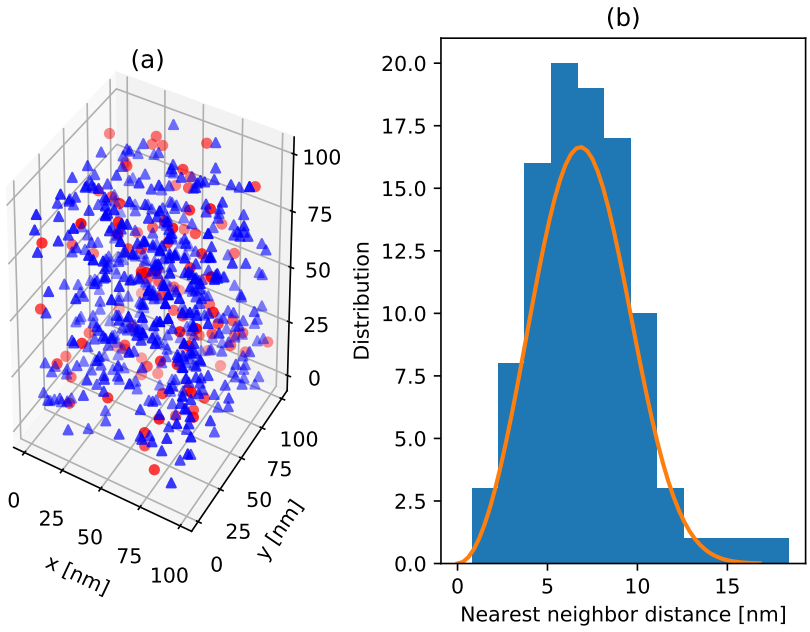

Fig. 4.1: (a) A cube with side $d = 100$ nm contains 100 electrons (circles) and 500 recombination centers (triangles). (b) Histogram of the distribution of nearest neighbor distances of electron-acceptor pairs $g(r)$ from the cube in (a). The solid line in (b) represents the analytical Eq.(**??**) for the distribution of nearest neighbors. For more details see Pagonis and Kulp [**?**].

Code 4.2: Time evolution of the nearest neighbors distribution

```
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
s=3e15                  # frequency factor
rho=1e-6                # rho-prime values 0.005-0.02
rc=0.0                  # for freshly irradiated samples, rc=0
times=(0,3.154e9,3.154e11,3.154e13)       # times in seconds
rprimes=np.arange(0,2.2,0.002)            # rprime=0-2.2
##### function to find distribution of distances ###
def findDistr(tim):
    return 3*(rprimes**2.0)*np.exp(-(rprimes**3.0))*\
    np.exp(-np.exp(-(rho**(-1/3))*rprimes)*s*tim)
distribs=[findDistr(x) for x in times]
plt.plot(rprimes,distribs[0],'-',c='black',linewidth=4,\
label=r't=0');
plt.plot(rprimes,distribs[1],'+-',c='r',label=r'10$^{2}$ a');
plt.plot(rprimes,distribs[2],'--',c='b',label=r'10$^{4}$ a');
plt.plot(rprimes,distribs[3],label=r'10$^{6}$ a');
plt.ylim(0,1.3);
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.ylabel('Nearest neighbor Distribution g(r$^{\'}$)');
plt.xlabel('Dimensionless distance r$^{\'}$');
plt.title('Time evolution of nearest neighbor distribution');
plt.tight_layout()
plt.show()
```

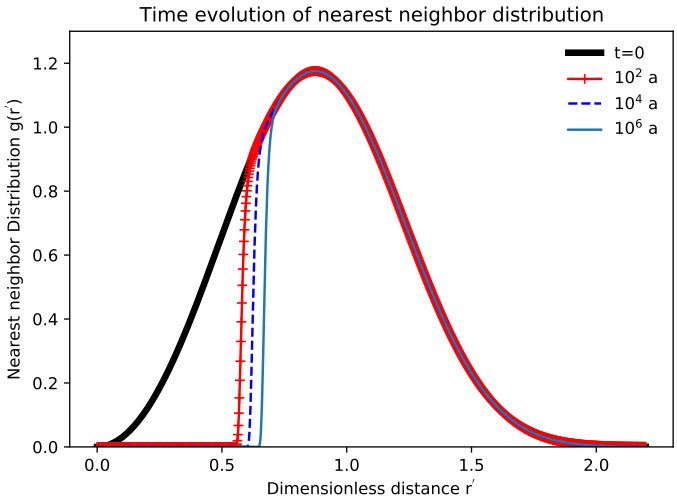

Fig. 4.2: Examples of the nearest neighbor distribution at different times $t = 0, 10^2, 10^4, 10^6$ a, based on Eq.(**??**). The solid black line represents the nearly symmetric distribution $g(r')$ at time $t = 0$. As time increases, the "tunneling front" is the almost vertical line which moves to the right, as more and more electrons are recombining at larger distances r'. For more examples see Pagonis et al. [**?**]

Code 4.3: Numerical evaluation of n(t) in GST model

```
#Example of summing exponentials to obtain n(t)in GST
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
### function to find n(r',t) at time t and at distance r'###
def nrprimet(tim,rprime):
    seff=s*np.exp(-(rho**(-1/3))*rprime)
    nt=3*(rprime**2.0)*np.exp(-(rprime**3.0))*\
    np.exp(-seff*tim)
    return nt*dr
s=3e15                          # GST
dr=0.01                         # interval in r'
rprimes=np.arange(0,2.2,dr)     # rprime=0-2.2
y=24*3600*365                   # year in s
times=10e3*y*np.linspace(0.01,2,30)
rho=1e-6                        # acceptor density
u=np.array([[nrprimet(x,rprimes[i]) for x in times]\
for i in range(len(rprimes))])
timesy=[i /y for i in times]
plt.plot(timesy,sum(u),'o',c='b',label=r'$\rho$'+"'"+'=\
1x10$^{-6}$');
ts=y*np.arange(0,2e4,10)
plt.plot(ts/y,np.exp(-rho*np.log(1.8*s*ts)**3.0));
plt.ylim(0.5,1);
plt.ylabel('Remaining charge  n(t)/no');
plt.xlabel('Elapsed time t [a]');
plt.title('Loss of charge in GST model');
rho=1.5e-6                       # acceptor density
u=np.array([[nrprimet(x,rprimes[i]) for x in times]\
for i in range(len(rprimes))])
plt.plot(timesy,sum(u),'^',c='r',label=\
```

```
r'$\rho$'+"'"+'=1.5x10$^{-6}$');
plt.plot(ts/y,np.exp(-rho*np.log(1.8*s*ts)**3.0));
leg = plt.legend()
plt.text(.95e4,.95,'Acceptor density');
leg.get_frame().set_linewidth(0.0)
plt.tight_layout()
plt.show()
```

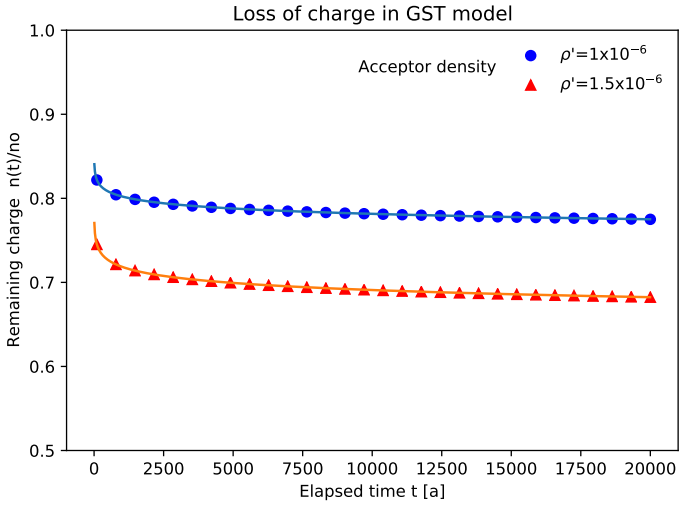

Fig. 4.3: Numerical integration of $n(r', t)$ from Eq.(**??**), to obtain the total concentration $n(t)$ of remaining trapped electrons at time t. The solid lines represent the analytical Eq.(**??**). This is a demonstration of the AF phenomenon in feldspars.

Code 4.4: Ground state tunneling: Remaining electrons n(t)

```
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
# Simulate Loss of charge due to ground state tunneling (GST)
z=1.8                    # constant in GST model
s=3e15                   # frequency factor
years=1000               # number of years elapsed
elapsedt=3.154e7*years   # change years to seconds
```

```
times=np.arange(1,elapsedt,1e6)          # times in seconds
rhos=[1e-6,5e-6,1e-5]                    # rho-prime values
##### function to find n(t) ###
def n(rho):
    return 100*np.exp(-rho*(np.log(z*s*times)**3.0))
ns=[n(x) for x in rhos]
plt.plot(times/3.154e7,ns[0],'-',c='black',\
linewidth=2,label=r'$\rho$'+"'"+'=10$^{-6}$');
plt.plot(times/3.154e7,ns[1],'-.',c='r',\
linewidth=2,label=r'$\rho$'+"'"+'=5x10$^{-6}$');
plt.plot(times/3.154e7,ns[2],'--',c='b',\
linewidth=2,label=r'$\rho$'+"'"+'=10$^{-5}$');
leg = plt.legend()
plt.ylim(0,120);
leg.get_frame().set_linewidth(0.0)
plt.xlabel('Elapsed time t [a]');
plt.ylabel('Remaining electrons [%]');
plt.title('Loss of charge in ground state tunneling');
plt.tight_layout()
plt.show()
```

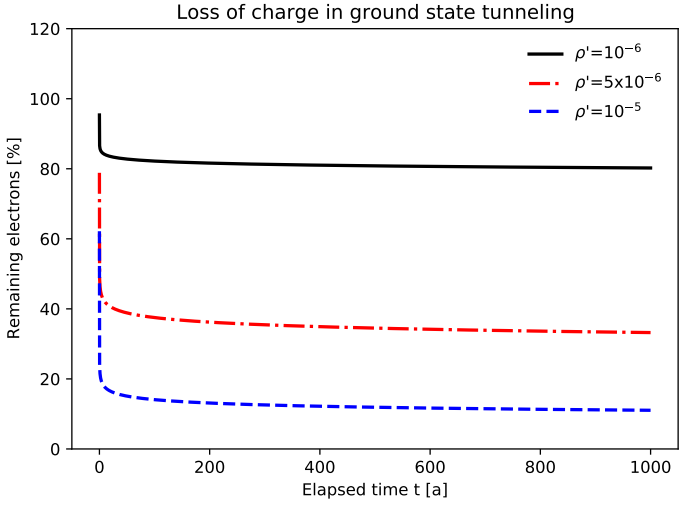

Fig. 4.4: Examples of the loss of charge due to ground state tunneling, for different dimensionless acceptor densities $\rho' = 10^{-6}, 5\times10^{-6}, 10^{-5}$, based on the Huntley Eq.(**??**). As ρ' increases, the rate of charge loss increases. As the elapsed time t increases, the initial fast loss of charge is followed by a "long tailed" decay, characteristic of quantum tunneling phenomena. See also Pagonis et al. [**?**]

Code 4.5: Numerical evaluation of n(t) in IGST model

```
#Example of summing exponentials to obtain n(t)in IGST model
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
### function to find n(r',t) at time t and at distance r'###
def nrprimet(tim,rprime,rho):
    seff=s*np.exp(-(rho**(-1/3))*rprime)
    expsoln=3*(rprime**2.0)*np.exp(-(rprime**3.0))*\
    X/(Do*seff+X)*(1-np.exp(-(X/Do+seff)*tim))
    return expsoln*dr
s=2e15                    # frequency factor
Do=1600                       # D0 in Gy
X=2.85/(1000*365*3600*24)     # natural dose rate 2.85 Gy/ka
dr=0.1                        # interval in r'
rprimes=np.arange(0,2.2,dr)     # rprime=0-2.2
y=24*3600*365                   # year in s
times=1e6*y*np.linspace(0,2,20)
timesy=[i /y for i in times]
rhos=[1e-6,2e-6,3e-6]           # rho-prime values 0.005-0.02
styls=['o','^','+']
for j in range(3):
    u=np.array([[nrprimet(x,rprimes[i],rhos[j]) for x in times]\
    for i in range(len(rprimes))])
    plt.plot(timesy,sum(u),styls[j],label=r'$\rho$='+str(rhos[j]));
    ts=y*np.arange(0,2.1e6,1e5)
    plt.plot(ts/y,(1-np.exp(-(X/Do)*ts))*np.exp(-rhos[j]*\
    np.log(Do*s/X)**3.0));
plt.ylim(0,1.2);
plt.ylabel('Percent of filled traps,  n(t)/N');
plt.xlabel('Elapsed time [a]');
plt.title('IGST model numerical integration');
```

```
leg = plt.legend()
plt.text(1.e6,1.12,'Acceptor density');
leg.get_frame().set_linewidth(0.0)
plt.tight_layout()
plt.show()
```

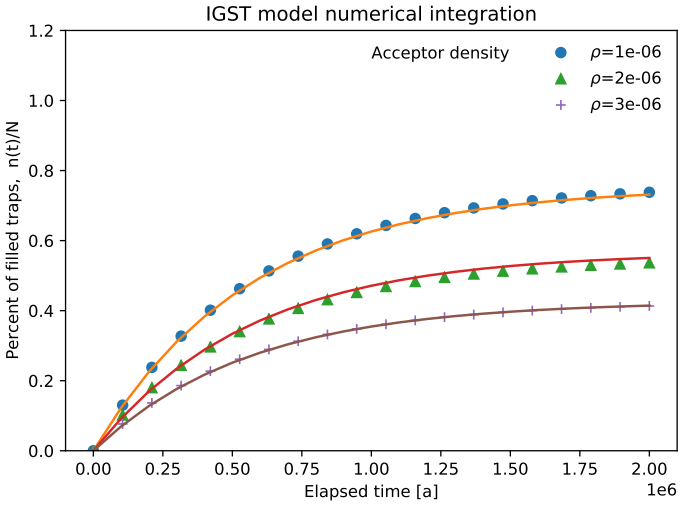

Fig. 4.5: Numerical integration of $n(r', t)$ from Eq.(**??**) in the IGST model, to obtain the total concentration $n(t)$ of remaining trapped electrons at time t. The solid lines represent the analytical Eq.(**??**).

Code 4.6: Simultaneous irradiation and anomalous fading in nature

```
# Simultaneous irradiation and anomalous fading in nature
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
s=3e15                  # frequency factor
Do=538                  # D0 in Gy
X=3/(1000*365*3600*24)  # natural dose rate 3 Gy/ka
years=1000              # number of years elapsed
elapsedt=3.154e7*years  # change years to seconds
times=np.arange(1,elapsedt,1e6)          # times in seconds
```

```
rhos=[1e-6,2e-6,3e-6]                    #rho-prime values
##### function to find n(t)
def n(rho):
    return (1-np.exp(-dose/Do))*np.exp(-rho*(np.log(Do*s/X)\
    **3.0))
dose=np.arange(1,3500,200)
ns=[n(x) for x in rhos]
plt.plot(dose,ns[0],'-',c='black',
linewidth=2,label=r'$\rho$'+"'"+'=1x10$^{-6}$');
plt.plot(dose,ns[1],'-.',c='r',
linewidth=2,label=r'$\rho$'+"'"+'=2x10$^{-6}$');
plt.plot(dose,ns[2],'--',c='b',
linewidth=2,label=r'$\rho$'+"'"+'=3x10$^{-6}$');
leg = plt.legend()
plt.ylim(0,1.20);
plt.text(1500,1.1,'Acceptor density');
leg.get_frame().set_linewidth(0.0)
plt.xlabel('Natural dose [Gy]');
plt.ylabel('L$_{FADED}$(D)');
plt.title('Simultaneous irradiation and anomalous fading');
plt.show()
```

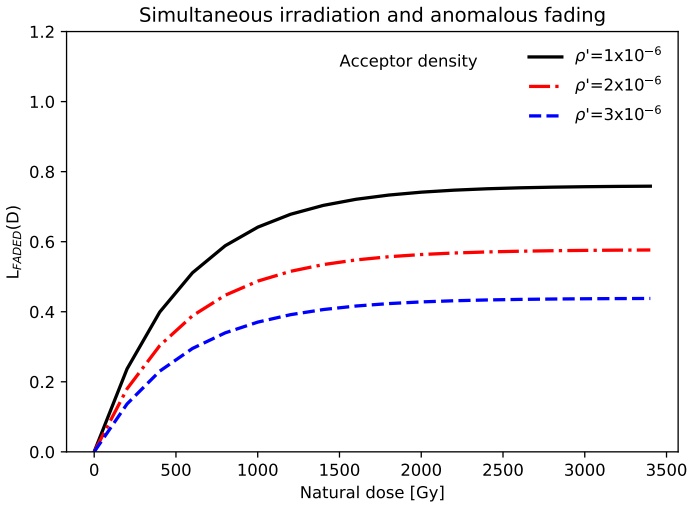

Fig. 4.6: Examples of the dose response of a faded sample in the IGST model. The faded signal L_{FADED} is proportional to the accumulated charge $n(t)$ during simultaneous irradiation and fading in nature, and is plotted here for three different dimensionless acceptor densities $\rho' = 10^{-6}$, 2×10^{-6}, 3×10^{-6}. As ρ' increases, the rate of charge accumulation and the corresponding saturation signal *decrease*. For more details, see Pagonis and Kitis [**?**].

Code 4.7: Example of summing partial TL curves

```
#Example of summing partial TL curves simplest
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()
```

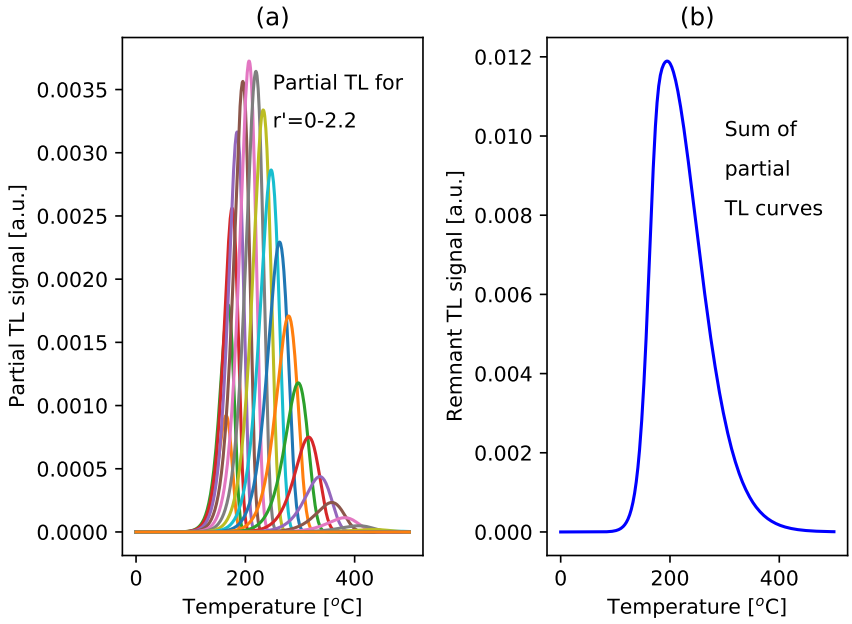

Fig. 4.7: (a) Simulation of the TL signal from an unfaded feldspar sample. (a) 22 partial first order TL glow curves which correspond to different distances r'; (b) The total TL signal from the sample is calculated as the sum of the partial first order TL glow curves in (a). The summation process results in the very broad TL peak characteristic of feldspars and apatites (Pagonis et al. [**?**]).

Code 4.8: Simulation of remnant TL after fading in nature

```
#Simulation of remnant TL after fading due to GST in nature
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
##### function to find distribution of distances in GST model
### after fading in nature for time tim
def findDistr(tim):
    return 3*(rprimes**2.0)*np.exp(-(rprimes**3.0))*\
    np.exp(-np.exp(-(rho**(-1/3))*rprimes)*stun*tim)
# find partial TL using the EST model
def partialTL(tmps,rprime,tim):
    seff=sEST*np.exp(-(rhoEST**(-1/3))*rprime)
    distr=3*(rprime**2.0)*np.exp(-(rprime**3.0))*\
    np.exp(-np.exp(-(rho**(-1/3))*rprime)*stun*tim)
    TLpartial=distr*\
    (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
# Here we use s and rho values for GST model
stun=3e15                   # frequency factor
rho=3e-6                # rho-prime value
timesNature=(3600*24*365,100*3600*24*365,1e4*3600*24*365)
# fading times in nature, in seconds
dr=0.02
rprimes=np.arange(0,2.2,dr)      # rprime=0-2.2
styls=['solid','dotted','dashed']
labls=[r't=1 a',r'10$^{2}$ a',r'10$^{4}$ a']
plt.subplot(1,2, 1);
for j in range(3):
    plt.plot(rprimes,findDistr(timesNature[j]),linestyle=\
    styls[j],label=labls[j]);
plt.ylim(0,1.3);
plt.xlim(0,2.3);
plt.ylabel('Trapped electron distribution n(r$^{\'}$,t)');
plt.xlabel('Dimensionless distance r$^{\'}$');
plt.title('(a)');
leg = plt.legend(loc='upper right');
leg.get_frame().set_linewidth(0.0)
## use different rhoEST and sEST for the EST model
kB,         sEST,        E ,    rhoEST,   hr=\
8.617e-5,  3.5e12,  1.45 ,    5e-3,   1
t = np.linspace(0, 500, 500)
```

```
temps=hr*t
plt.subplot(1,2, 2);
plt.title('(b)');
for j in range(3):
    u=np.array([[partialTL(x,rprimes[i],timesNature[j]) for\
    x in temps] for i in range(len(rprimes))])
    plt.plot(temps,sum(u),linestyle=styls[j],label=labls[j]);
plt.ylabel('Remnant TL signal [a.u.]');
plt.xlabel(r'Temperature [$^{o}$C]');
plt.xlim(150,500);
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.tight_layout()
plt.show()
```

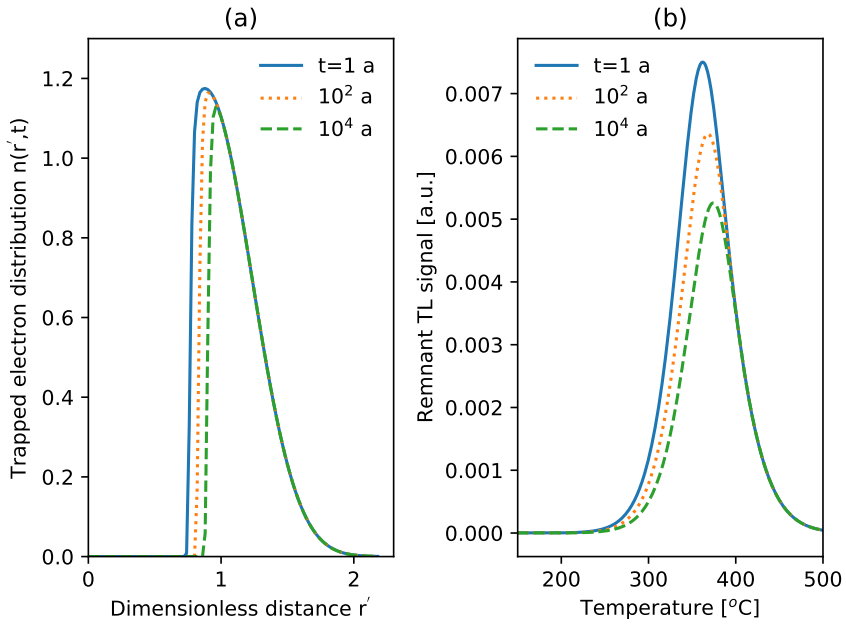

Fig. 4.8: Simulation of anomalous fading in nature, followed by measurements of the remnant-TL signals in the laboratory. (a) The nearest neighbor distributions at the end of the anomalous fading periods t_{AF} =1, 10^2, 10^4 a. This first part of the simulation is based on the GST model. (b) The corresponding remnant-TL signals measured in the laboratory. This second part of the simulation is based on the EST model.

Code 4.9: Dose response curve n(t) in TA-EST model

```
# Dose response n(t) in the TA-EST model by Brown et al.
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
##### function to find distribution of distances ###
def findDistr(tim):
    P=Po*np.exp(-(rho**(-1/3))*rprimes)
    Peff=(P*s)*np.exp(-E/(kB*(273+Tirr)))/(P+s)
    return 3*(rprimes**2.0)*np.exp(-(rprimes**3.0))*\
    X/(Do*Peff+X)*(1-np.exp(-(X/Do+Peff)*tim))
Tirr=-4                   # irradiation temeprature
Po=2e15                   # tunneling frequency factor
s=2e15                    # thermal frequency factor
rho=1e-3                  # rho-prime values 0.005-0.02
Do=1600                   # D0 in Gy
yr=365*3600*24            # year in seconds
X=2.85/(1000*yr)          # natural dose rate 2.85 Gy/ka
times=[yr*1e4,yr*1e5,yr*.2e6,yr*.4e6,yr*.7e6,yr*1e6] # time (s)
t=[times[x]/(1e6*yr) for x in range(6)]
kB, E =8.617e-5, 1.3
dr=0.05
rprimes=np.arange(0,2.2,dr)    # rprime=0-2.2
distribs=[findDistr(x) for x in times]
plt.subplot(1,2, 1);
labls=[str(x) for x in t]
for j in range(6):
    plt.plot(rprimes,distribs[j],'-',label=labls[j]);
plt.text(1.55,.32,'Irradiation');
plt.text(1.55,.27,'time in Ma');
plt.xlim(0,2.4);
plt.ylabel('Trapped electrons distribution n(r$^{\'}$,t)/N');
plt.xlabel('Distance r$^{\'}$');
plt.title('(a)');
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.subplot(1,2, 2);
sums=[sum(distribs[x])*dr for x in range(6)]
plt.plot(t,sums,'o-');
plt.title('(b)');
plt.ylabel('Total trapped electrons n(t)/N');
```

```
plt.xlabel('Irradiation time [Ma]');
plt.text(.33,.16,'TA-EST model');
plt.tight_layout()
plt.show()
```

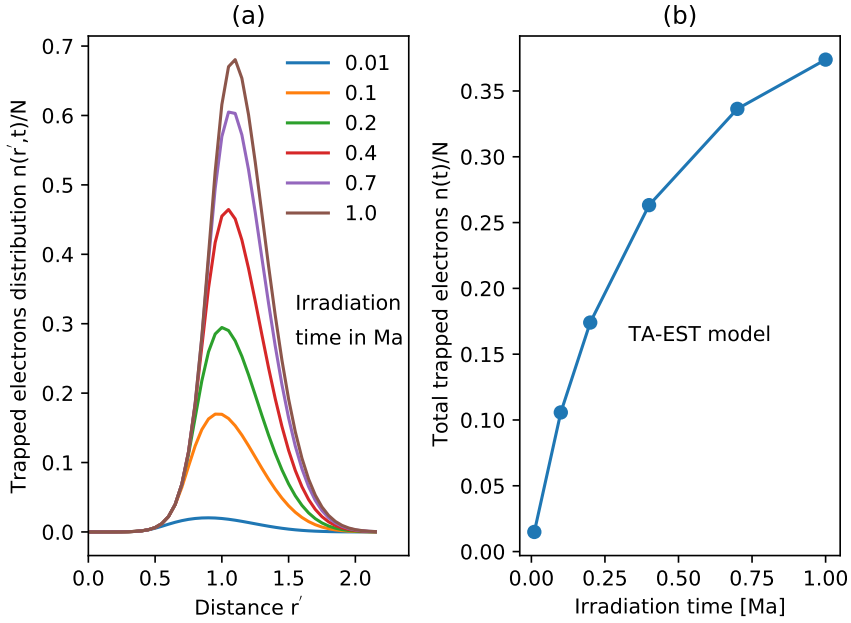

Fig. 4.9: (a) Distribution of trapped electrons $n(r', t)$ in the TA-EST model, for irradiation times t =10^4-10^6 a. (b) Total remaining charge $n(t)$ as a function of irradiation time t.