Chapter 14
DOSE RESPONSE OF DOSIMETRIC MATERIALS: MODELS

Code 14.1: Irradiation: OTOR, Lambert analytical solution

```
# GOT MODEL- code for IRRADIATION Lambert solution
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from scipy.special import lambertw
# Numerical parameters for GOT irradiation model.
N,     R, X,    n0 =\
1E10, .1, 1e10, 0
t = np.linspace(0, 100, 100)
# The GOT model irradiation differential equation.
def deriv(y, t):
    n = y
    dndt=   (N-n) * X*R / ((N - n) * R + n )
    return dndt
y0 = n0
ret = odeint(deriv, y0, t)
n= ret.flatten()
plt.plot(t, n, 'r^',  lw=2, label='GOT model');
ret = odeint(deriv, y0, t)
#analytical n(t)
Dc=N/R
u=N*(1+np.real(lambertw((R-1)*np.exp((R-1)-\
(t*X/np.abs(Dc))))/(1-R)))
plt.plot(t, u,label='PKC equation');
```

```
leg = plt.legend(loc='right')
leg.get_frame().set_linewidth(0.0)
plt.ylabel('Trap filling ratio, n/N');
plt.ylim(0,1.2e10);
plt.xlim(0,100);
plt.title('Dose response in GOT model, with PKC eqt');
plt.text(20,.6e10,'GOT model');
plt.text(20,.5e10,'Analytical solution');
plt.text(20,.4e10,'with Lambert');
plt.xlabel('Time [s]');
plt.tight_layout()
plt.show()
```

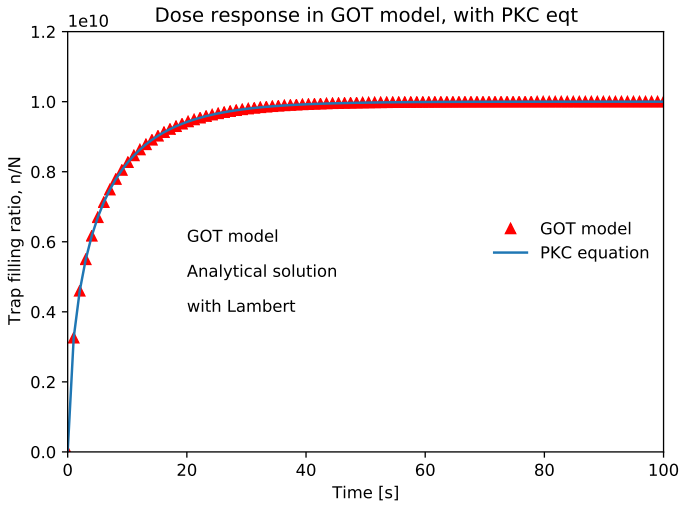

Fig. 14.1: Dose response in the OTOR model, using the Lambert analytical solution PKC Eq.(**??**). The analytical solution (solid line) is compared with the numerical solution of the differential Eq.(**??**) shown as triangles.

Code 14.2: Plots of the PKC equation

```
#plot dose response using PKC equation for different R, Dc
import numpy as np
```

```
import matplotlib.pyplot as plt
from scipy.special import lambertw
import warnings
warnings.filterwarnings("ignore")
def PKC(R,Dc):
    u=(1+np.real(lambertw((R-1)*np.exp((R-1)-\
    (D/np.abs(Dc))))/(1-R)))
    plt.plot(D,u/max(u),symbs[j-1], linewidth=2,
    label=labls[j-1]);
Dc=20
D = np.linspace(0, 100, 100)
Rs=[.01,.5,.9]
labls=["R="+str(x) for x in Rs]
symbs=['+-','^-','o-']
plt.subplot(1,2,1);
for j in range(1,4):
    PKC(Rs[j-1],Dc)
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.xlabel('Dose [Gy]');
plt.text(45,.6,'PKC equation');
plt.ylabel('n(D)/N');
plt.title('(a)');
plt.subplot(1,2,2);
Dcs=[5,10,20]
labls=['Dc='+str(x)+' Gy' for x in Dcs]
for j in range(1,4):
    PKC(0.01,Dcs[j-1])
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.xlabel('Dose [Gy]');
plt.ylabel('n(D)/N');
plt.title('(b)');
plt.tight_layout()
plt.show()
```

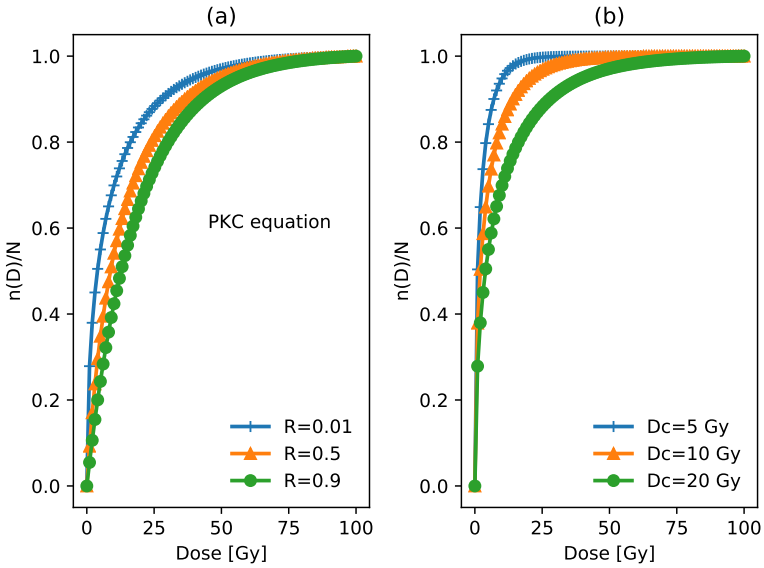

Fig. 14.2: (a) Plot of the PKC Eq.(**??**) for 3 values of the retrapping ratio R = 0.01, 0.5, 0.9 and a fixed characteristic dose D_c =20 Gy. As the value of R increases, the dose response curve takes longer to reach saturation, due to retrapping (b) Plot of the PKC equation for 3 values of D_c =5, 10, 20 Gy and a fixed R =0.01. As the value of D_c increases, the dose response curve takes longer to reach saturation.

Code 14.3: Plots of the PKC-S equation

```
#plot dose response  PKC-S equation for different beta, B
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import lambertw
import warnings
warnings.filterwarnings("ignore")
def PKCS(B,beta):
    u=np.real(lambertw(B*np.exp(np.abs(B)-(D/Dc))))/B
    u=(1-(u**beta))
    plt.plot(D,u,symbs[j-1], linewidth=2,
    label=labls[j-1]);
```

```
    plt.yscale("log");
    plt.xscale("log");
B, Dc= 2, 2
D = np.linspace(.1, 40, 100)
betas=[.01,.1,.9]
labls=[r'$\beta$='+str(x) for x in betas]
symbs=['+-','^-','o-']
plt.subplot(1,2,1);
for j in range(1,4):
    PKCS(B,betas[j-1])
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.xlabel('Dose [Gy]');
plt.text(.1,.8,'PKC-S equation')
# plt.text(45,.6,'PKC equation');
plt.ylabel('n(D)/N');
plt.title('(a)');
plt.subplot(1,2,2);
Bs=[.1,1,2]
labls=['B='+str(x) for x in Bs]
for j in range(1,4):
    PKCS(Bs[j-1],.01)
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.xlabel('Dose [Gy]');
plt.ylabel('n(D)/N');
plt.title('(b)');
plt.tight_layout()
plt.show()
```

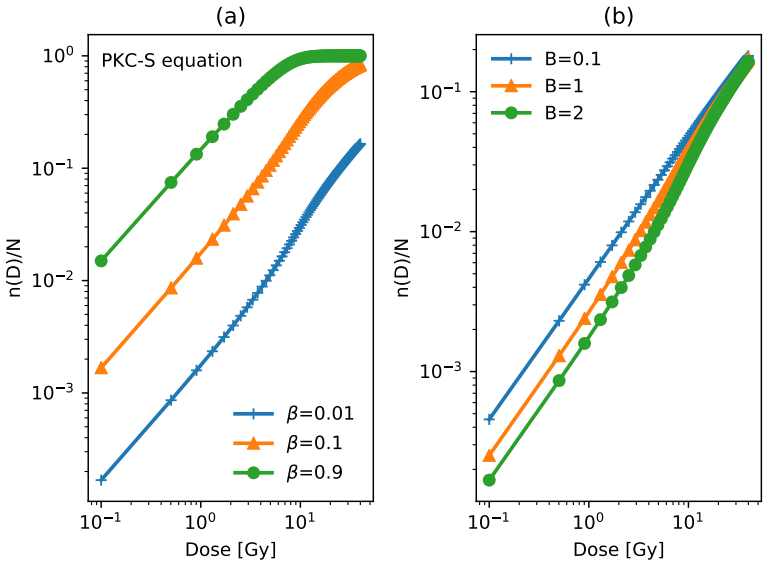

Fig. 14.3: (a) Plot of the PKC-S Eq.(**??**) for 3 values of the constant β = 0.01, 0.1, 0.9 and a fixed constant B =2. As the value of β increases, the amount of superlinearity decreases. (b) Plot of the PKC-S equation for B =0.1, 1, 2 and a fixed β =0.01. As the value of B increases, the amount of superlinearity increases. For more details see Pagonis et al. [1]

Code 14.4: Plots of the supralinearity index f(D)

```
#plot coefficient f(D) for different beta, B
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import lambertw
import warnings
warnings.filterwarnings("ignore")
def PKCS(B,beta):
    u=np.real(lambertw(B*np.exp(np.abs(B)-(D/Dc))))/B
    u=(1-(u**beta))/D
    u1=np.real(lambertw(B*np.exp(np.abs(B)-(.1/Dc))))/B
    u1=(1-(u1**beta))/.1
    plt.plot(D,u/u1,symbs[j-1], linewidth=2,
```

```
    label=labls[j-1]);
    plt.xscale("log");
B, Dc= 2, 2
D = np.linspace(.1, 40, 40)
betas=[.1,.3,.9]
labls=[r'$\beta$='+str(x) for x in betas]
symbs=['+-','^-','o-']
plt.subplot(1,2,1);
for j in range(1,4):
    PKCS(B,betas[j-1])
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.xlabel('Dose [Gy]');
plt.ylabel('f(D)');
plt.title('(a)');
plt.subplot(1,2,2);
Bs=[1,2,3]
labls=['B='+str(x) for x in Bs]
for j in range(1,4):
    PKCS(Bs[j-1],.1)
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.ylim(0,3);
plt.xlabel('Dose [Gy]');
plt.ylabel('f(D)');
plt.title('(b)');
plt.tight_layout()
plt.show()
```

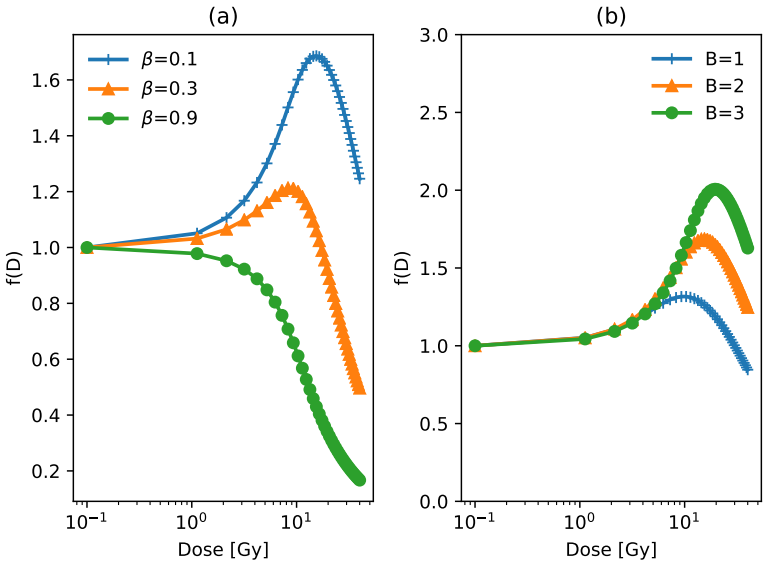

Fig. 14.4: (a) Plot of Eq.(**??**) for 3 values of the constant $\beta = 0.1, 0.3, 0.9$ and a fixed constant B =2. As the value of β increases, the value of $f(D)$ and the amount of superlinearity decreases. (b) Plot of Eq.(**??** for B =1, 2, 3 and a fixed β =0.1. As the value of B increases, the value of $f(D)$ and the amount of superlinearity increases. For more details see Pagonis et al. [1]

References

1. 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