Free energy between two skewed cylinders (CG-10 in water). Full retarded result, function of separation ℓ and angle θ

Equation 12: $G(\ell,\theta) = -\frac{(\pi R_1^2)(\pi R_2^2)}{2\pi\ \ell^4 \sin\theta}\left(\mathscr{A}^{(0)}(\ell) + \mathscr{A}^{(2)}(\ell)\cos 2\theta\right)$

$G(\ell,\theta) = -\frac{k_B T}{64\pi}\frac{\pi^2 R_1^2 R_2^2}{\ell^4 \sin\theta}\sum_{n=0}^{\infty}{}' \Delta_{1,\parallel}\Delta_{2,\parallel}\ p_n^4\ \int_0^\infty t dt\ \frac{e^{-2p_n\sqrt{t^2+1}}}{(t^2+1)}\tilde{g}(t, a_1(i\omega_n), a_2(i\omega_n), \theta)$,

with $\tilde{g}(t,a_1,a_2,\theta) = 2\left[(1+3a_1)(1+3a_2)t^4 + 2(1+2a_1+2a_2+3a_1a_2)t^2 + 2(1+a_1)(1+a_2)\right] +$
$+ (1-a_1)(1-a_2)(t^2+2)^2\cos 2\theta.$

/usr/bin/python

```
import numpy as np
import scipy.optimize as opt
from scipy.integrate import trapz
import matplotlib.pyplot as pl
from matplotlib import axis as ax
# use pyreport -l file.py
from pylab import show
from matplotlib.ticker import MultipleLocator
from mpl_toolkits.mplot3d import Axes3D
from pylab import pause


eiz_x = np.loadtxt('data/eiz_x_output_eV.txt') #perpendicular, radial

eiz_y = np.loadtxt('data/eiz_y_output_eV.txt')
eiz_z = np.loadtxt('data/eiz_z_output_eV.txt') # parallel,axial

eiz_w = np.loadtxt('data/eiz_w_output_eV.txt') # water as intervening medium


eiz_w[0] = eiz_w[1] #NOTE: there is a jump from first val down to second val


r_1 = 0.5e-9
r_2 = 0.5e-9
c = 2.99e8 # in m/s

#T = 1.
#kb = 1.
# at RT, 1 kT = 4.11e-21 J
# 1 eV = 1.602e-19 J = 0.016 zJ
# h_bar_eV = 6.5821e-16 eVs
T = 297
# h_bar = 1. #1.0546e-34 #in Js
kb = 8.6173e-5  # in eV/K

#kb  = 1.3807e-23 # in J/K
# NOTES:
# z_n_eV = (2*pi*kT/h_bar)n
#        = (0.159 eV) / (6.5821e-16 eVs)
#        = n*2.411e14 rad/s
# z_n_J = (2*pi*kT/h_bar)n
#        = (1.3807e-23 J/K) / (1.0546e-34 Js))*n
#        = n*2.411e14 rad/s
#coeff = 0.159 # in eV w/o 1/h_bar
coeff = 2.411e14 # in rad/s


ns = np.arange(0.,500.)
z = ns * coeff
```

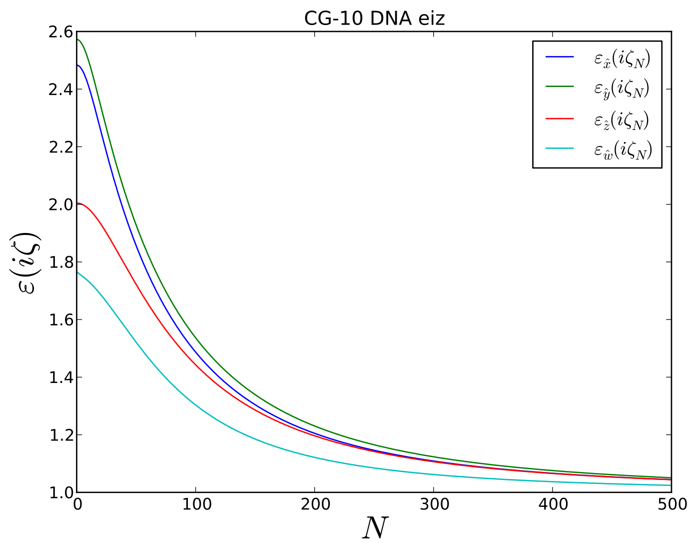
```
ls = np.linspace(1.0e-9, 7.0e-9, 10)
#ls = np.linspace(1.0e-9, 7.0e-8, 50)#this one has been working fine
#ls = np.linspace(1.0e-8, 7.0e-8, 50)
#thetas = np.linspace((1./22)*np.pi,(1./2)*np.pi,50) #this one has been working
    fine
#ls = np.linspace(1.0e-8, 7.0e-8, 50)
thetas = np.linspace((0.0001)*np.pi,(1./2)*np.pi,10)
#thetas = np.linspace((1./8)*np.pi,(1./2)*np.pi,50)

def Aiz(perp, par,med):
        return (2.0*(perp-med)*med)/((perp+med)*(par-med))
def ys(a,time,eizw,L, N):
        term0 = np.log( time    / (time*time+1.0)                   )
        term1 = np.log( time**4 * 2.0*(1. + 3.*a)*(1.+3.*a)         )
        term2 = np.log( time**2 * 4.0*(1. + 2.0*a+2.0*a+3.0*a*a))
        term3 = np.log(           4.0*(1. + a)*(1.0 + a)            )
        term4 = (-2.0 * np.sqrt(eizw)* L * coeff * N / c * np.sqrt(time*time +
            1.0))
        return np.exp(term0 + term1 + term2 + term3 + term4)#* term5

def y_2s(a,time,eizw, L, N):
        term0 = np.log(time     / (time*time+1.0)                            )
        term1 = np.log((1.- a)*(1.- a)*(time * time  + 2.0)*(time * time + 2.0))
        term2 = (-2.0 * np.sqrt(eizw)* L * coeff * N / c * np.sqrt(time*time +
            1.0))
        return np.exp(term0 + term1 + term2) #* term3

def As(eizz,eizw,L,N,Y):
        term0 = 1.0 #(1.0/32)*kb*T

        term1 = ((eizz-eizw)/eizw)*((eizz-eizw)/eizw)
        term2 = eizw *eizw * (coeff*N)**4 * L**4 / (c**4) #NOTE: took out 1/c^4
            for both A's

        term3 = Y
        return term0 * term1 * term2 * term3
def A_2s(eizz,eizw, L , N ,Y):
        term0 = 1.0 #(1.0/32)*kb*T

        term1 = ((eizz-eizw)/eizw)*((eizz-eizw)/eizw)
        term2 = eizw *eizw * (coeff*N)**4 * L**4 / (c**4)
        term3 = Y
        return term0 * term1 * term2 * term3

dt = 1000000000000000e-15
ts = np.arange(1e-27,1000000000000000000e-15,dt)

A   = np.zeros(shape=(len(ns),len(ls)))
A_2 = np.zeros(shape=(len(ns),len(thetas),len(ls)))
aiz = []
sum_A = np.empty(len(ls))
sum_A_2 = np.zeros(shape=(len(thetas),len(ls)))
#sum_A_2 = np.empty(len(ls))
EL = np.zeros(len(ls))
G_l_t_dt = np.zeros(shape=(len(thetas),len(ls)))

aiz = Aiz(eiz_x,eiz_z, eiz_w) # of length = len(ns)


#for j,n in enumerate(ns):
```

```
#       print "on n=%d of %d"%(j,len(ns))
#       for k,l in enumerate(ls):
#               # Integrand:
#               y_arg   = ys(aiz[j],ts,eiz_w[j],l,n)
#               y_2_arg = y_2s(aiz[j],ts,eiz_w[j],l,n)
#               # Integral:
#               y   = trapz(y_arg,ts,dt)
#               y_2 = trapz(y_2_arg,ts,dt)
#               A[j,k]   = As(eiz_z[j],eiz_w[j],l,n,y)
#               for i, theta in enumerate(thetas):
#                       A_2[j,k,i] = A_2s(eiz_z[j],eiz_w[j],l,n,y_2) * np.cos
    (2.0*theta)
#sum_A = np.sum(A,axis=0)
#sum_A_2 = np.sum(A_2,axis=0)
#pl.figure()
for i, theta in enumerate(thetas):
        print 'i,theta = ', (i,theta)
        for k,length in enumerate(ls):
                for j,n in enumerate(ns):
                        #print "on n=%d of %d"%(j,len(ns))
                        # Integrand:
                        y_arg   = ys(aiz[j],ts,eiz_w[j],length,n)
                        y_2_arg = y_2s(aiz[j],ts,eiz_w[j],length,n)
                        # Integral:
                        y   = trapz(y_arg,ts,dt)
                        y_2 = trapz(y_2_arg,ts,dt)
                        A[j,k]   = As(eiz_z[j],eiz_w[j],length,n,y)
                        A_2[j,i,k] = A_2s(eiz_z[j],eiz_w[j],length,n,y_2) * np.
                            cos(2.0*theta)
                        A[0,k] = (1./2)*A[0,k]
                        A_2[0,i,k] = (1./2)*A_2[0,i,k]
                sum_A = np.sum(A,axis=0)
                #print 'shape sum_A = ', np.shape(sum_A)
                sum_A_2 = np.sum(A_2,axis=0)
                #print 'shape sum_A_2 = ', np.shape(sum_A_2)
                #sys.exit()
                EL[k] = 1./(length*length*length*length)
                G_l_t_dt[i,k] = 1.602e-19 *6.5821e-16 *kb * T * (1./32) * EL[k]*
                    np.pi*r_1*r_1*r_2*r_2*(sum_A[k] + sum_A_2[i,k])/(2.0*np.sin(
                    theta))# (1e21)*

                labels = 'theta = %.1f' %(theta)
```

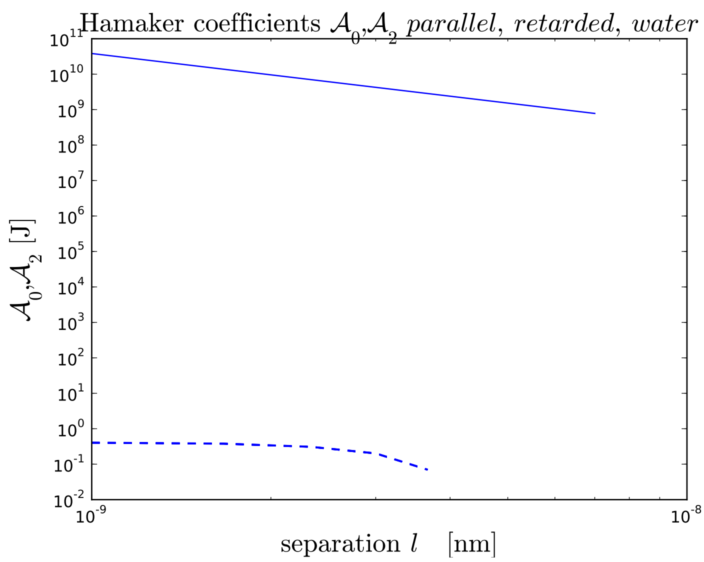
```
i,theta =  (0, 0.00031415926535897931)
i,theta =  (1, 0.17481217787975206)
i,theta =  (2, 0.34931019649414513)
i,theta =  (3, 0.52380821510853814)
i,theta =  (4, 0.69830623372293121)
i,theta =  (5, 0.87280425233732428)
i,theta =  (6, 1.0473022709517175)
i,theta =  (7, 1.2218002895661106)
i,theta =  (8, 1.3962983081805036)
i,theta =  (9, 1.5707963267948966)
```

```
                #print 'theta = %.1f, length = %.1f, G = %s' %(i,k,G_l_t_dt[k,i]
                    )
#       pl.plot(ls, G_l_t_dt, label = labels)
#pl.show()

# 1 eV = 1.602e-19 J = 0.016 zJ
# h_bar_eV = 6.5821e-16 eVs
```

```
pl.figure()
pl.plot(ns,eiz_x, color = 'b', label = r'$\varepsilon_{\hat{x}}(i\zeta_{N})$')
pl.plot(ns,eiz_y, color = 'g', label = r'$\varepsilon_{\hat{y}}(i\zeta_{N})$')
pl.plot(ns,eiz_z, color = 'r', label = r'$\varepsilon_{\hat{z}}(i\zeta_{N})$')
pl.plot(ns,eiz_w, color = 'c', label = r'$\varepsilon_{\hat{w}}(i\zeta_{N})$')
pl.xlabel(r'$N$', size = 24)
pl.ylabel(r'$\varepsilon(i\zeta)$', size = 24)
pl.legend()
pl.title(r'CG-10 DNA eiz')
show()
```

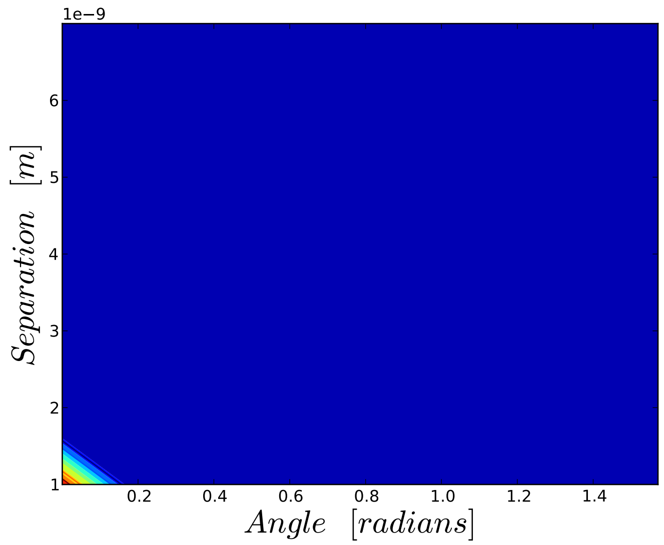



```

pl.figure()
pl.loglog(ls,sum_A,'b-')
pl.loglog(ls,sum_A_2,'b--')
pl.xlabel(r'$\mathrm{separation}\,\it{l}\,\,\,\rm{[nm]}$', size = 20)
pl.ylabel(r'$\mathrm{\mathcal{A_{0},A_{2}}\,[J]}$', size = 20)
pl.title(r'$\mathrm{Hamaker \, coefficients \,\mathcal{A_{0},A_{2}}}\, parallel
    ,\,retarded,\,water$', size = 20)
show()
```



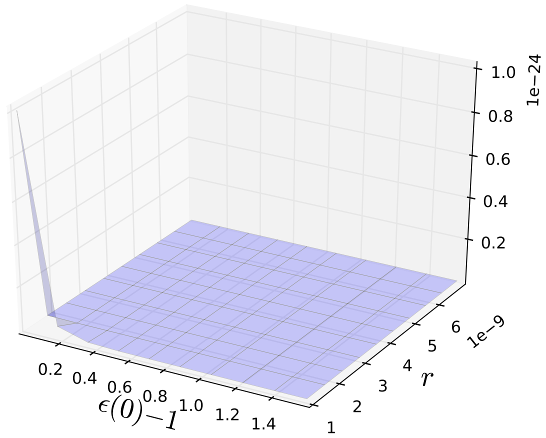

```

#G_l_t_dt[G_l_t_dt>200e-20]= np.nan #NOTE: remove me later
#G_l_t_dt[G_l_t_dt<200e-25]= np.nan #NOTE: remove me later
```

```
# CONTOUR PLOT:
X,Y = np.meshgrid(thetas, ls)
pl.figure()
pl.contourf(X, Y, G_l_t_dt, 10)#, cmap = cm.hot())

CS = pl.contour(X,Y,G_l_t_dt)#, levels = np.linspace(1e-1,1e10,10))

pl.clabel(CS, inline =1,fmt = '%1.5f', fontsize = 18,color = 'k')#, manual = man_loc)

pl.xlabel(r'$Angle\,\,[radians]$', size = 24)
pl.ylabel(r'$Separation\,\,[m]$', size = 24)
#cbar = pl.colorbar(CS, shrink = 0.8, extend = 'both')
#cbar.ax.set_ylabel(r'$G(\mathcal{l},\theta)\,\,[zJ]$', size = 24)
#cbar.add_lines(CS)
##pl.axis([0,1.0,0,1.0])
#pl.grid()
show()
```

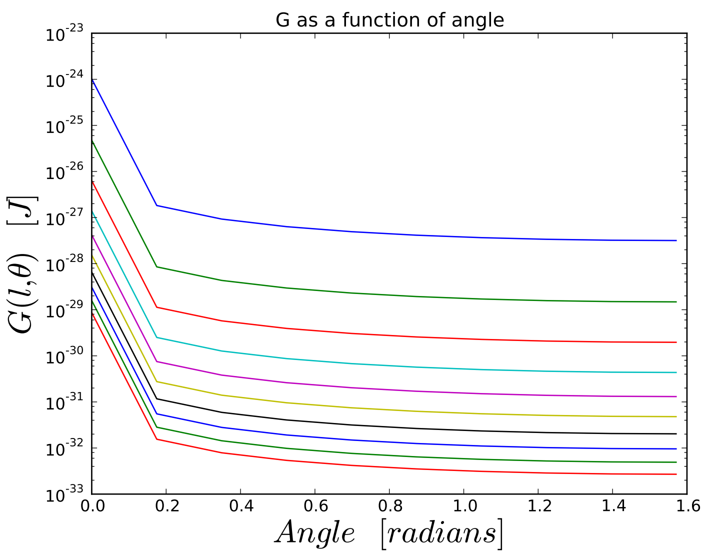



```

fig = pl.figure()
ax = fig.gca(projection = '3d')
#ax.text(-7, 6, 0.7, r'$\zeta/\omega_{0}$', zdir = (-1,1,-3), size = 21)
surf = ax.plot_surface(X,Y, G_l_t_dt, rstride = 1, cstride = 1,alpha = 0.2, linewidth = 0.3)#edgecolor = 'none',antialiased = True, shade = False, norm = norm, linewidth = 0.3)

#surf = ax.plot_surface(X,Y, G_l_t_dt, rstride = 20, cstride = 20,alpha = 0.2)#, cmap = cm.gnuplot, linewidth = 0.5)#gray)#coolwarm)#bone)#hot, linewidth = 0.01, antialiased = True, shade = False)# True)#, cmap = hot()
#colorbar(surf)
#cbar.ax.set_ylabel(r'$\frac{\xi}{\omega_{0}}$', size = 24)
#cset = ax.contour(X,Y,h, zdir = 'z', offset = 0, cmap = cm.jet)
#cset = ax.contour(X,Y,h, zdir = 'x', offset = 5, cmap = cm.jet)
#cset = ax.contourf(X,Y,h, zdir = 'y', offset = 6, cmap = cm.jet)# puts plot of max xi vs discrete r values at r=0 plane
#ax.view_init(elev = 19, azim = -112)
#zlabel(r'$\xi/\omega_{0}$', size = 21)
#ylabel(r'$r$', size = 24)
#xlabel(r'$(\epsilon(0) -1)$', size = 24)
#text = Axes.text(self, x, y, s, **kwargs)
#art3d.text_2d_to_3d(text, z, zdir)
```

```
#return text
#pl.text(6,0, 0, r'$\xi/\omega_{0}$',size = 21 ,rotation = 'horizontal')
#ax.text(r'$\xi/\omega_{0}$',6,0, 0, size = 21 ,rotation = 'horizontal')
#ax.set_zlabel(r'$\xi/\omega_{0}$',size = 21 ,rotation = 'horizontal' )
ax.set_xlabel(r'$\epsilon(0)-1$', size = 21)
ax.set_ylabel(r'$r$', size = 22)
show()
```

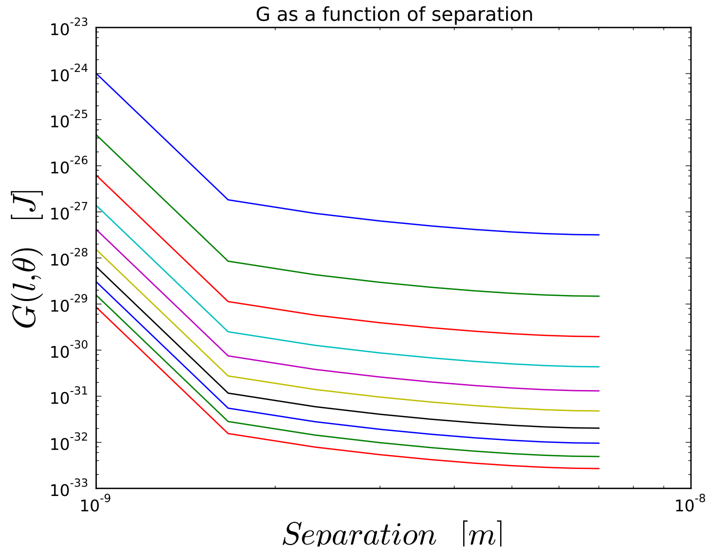



```

pl.figure()
pl.semilogy(thetas, G_l_t_dt)
pl.xlabel(r'$Angle\,\,[radians]$', size = 24)
pl.ylabel(r'$G(l,\theta)\,\,[J]$', size = 24)
#pl.axis([(1./25)*np.pi,(3./4)*np.pi,105,135])
pl.title('G as a function of angle')
show()
```

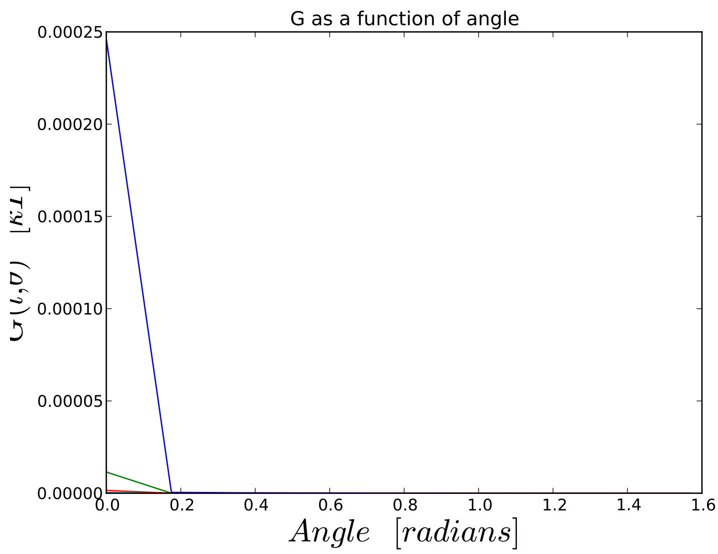



```

pl.figure()
pl.loglog(ls, G_l_t_dt)
pl.xlabel(r'$Separation\,\,[m]$', size = 24)
pl.ylabel(r'$G(l,\theta)\,\,[J]$', size = 24)
#pl.axis([1.5e-9, 6.5e-8,100,145])
pl.title('G as a function of separation')
```

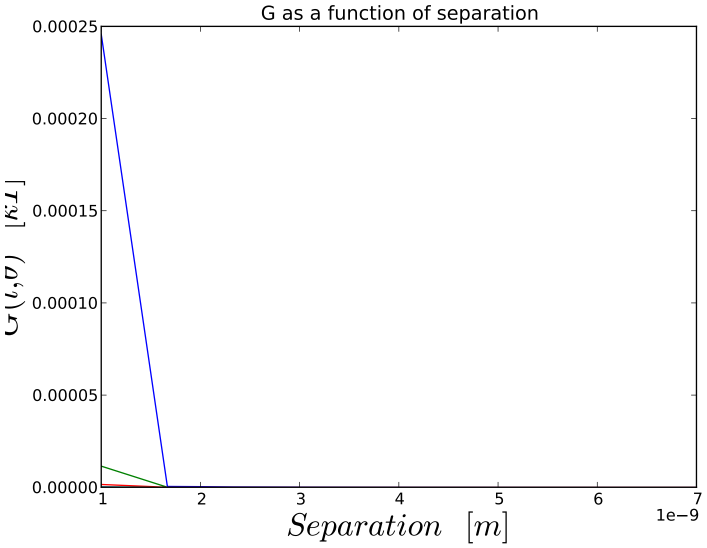
```
show()
```




```

pl.figure()
pl.plot(thetas, (1./(4.11e-21))*G_l_t_dt)
pl.xlabel(r'$Angle\,\,[radians]$', size = 24)
pl.ylabel(r'$G(l,\theta)\,\,[kT]$', size = 24)
#pl.axis([(1./25)*np.pi,(3./4)*np.pi,105,135])
pl.title('G as a function of angle')
show()
```




```

pl.figure()
pl.plot(ls, (1./(4.11e-21))*G_l_t_dt)
pl.xlabel(r'$Separation\,\,[m]$', size = 24)
pl.ylabel(r'$G(l,\theta)\,\,[kT]$', size = 24)
#pl.axis([1.5e-9, 6.5e-8,100,145])
pl.title('G as a function of separation')
show()
```


G as a function of separation
0.00025
0.00020
0.00015
0.00010
0.00005
0.00000
1
2
3
4
5
6
7
1e−9
Separation [m]