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

Numerically solve:

Equation 27: $g(\ell) = -\frac{k_BT}{32}\frac{R_1^2R_2^2}{\ell^5}\sum_{n=0}^{\infty}{}'\Delta_{1,\parallel}\Delta_{2,\parallel}\int_1^{+\infty}\frac{dy}{\sqrt{y^2-1}}\int_0^{\infty}udu\,\frac{e^{-2y\sqrt{u^2+p_n^2}}}{(u^2+p_n^2)^{1/2}}\,h(a_1(i\omega_n),a_2(i\omega_n),u,p_n^2)$,

and

$h(a_1(i\omega_n),a_2(i\omega_n),u,p_n^2) = 2\left[(1+3a_1)(1+3a_2)u^4+2(1+2a_1+2a_2+3a_1a_2)u^2p_n^2+2(1+a_1)(1+a_2)p_n^4\right]+(1-a_1)(1-a_2)(u^2+2p_n^2)^2$.

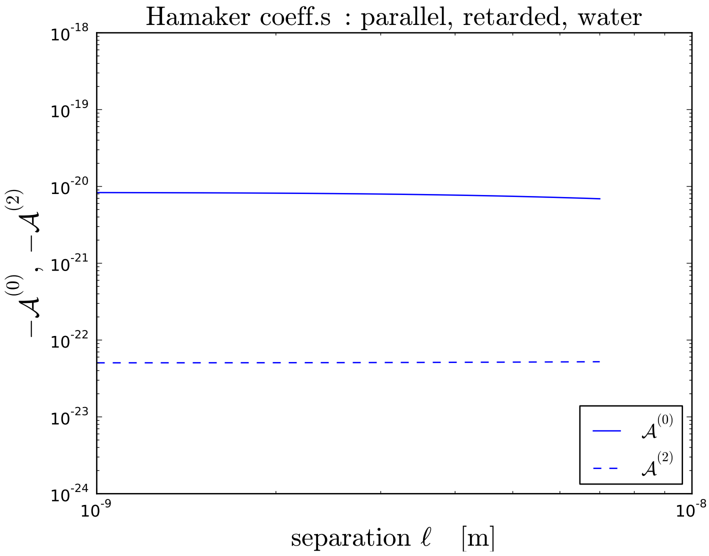
/usr/bin/python

```
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
from matplotlib.backends.backend_pdf import PdfPages
pp = PdfPages('plots/par_ret_water/par_ret_water.pdf')

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 = 1.0 + np.zeros(len(eiz_z))
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 = 297
kb  = 1.3807e-23 # in J/K

coeff = 2.411e14 # in rad/s

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

ns = np.arange(0.,500.)
z = ns * coeff
ls = np.linspace(1.0e-9, 7.0e-9, 25)
thetas = np.linspace((0.0001)*np.pi,(1./2)*np.pi,25)
dt = 1.0
```

```
ts = np.arange(1.0,10000.,dt)

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

def y_2s(a,time,eizw, L, N):
        term0 = (time    / (time*time+1.0)                              )
        term1 = ((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))
        #print 'y_2s term0', term0
        #print 'y_2s term1', term1
        #print 'y_2s term2', term2
        #print '----'
        return term0 * term1* np.exp(term2) #* term3

def As(eizz,eizw,L,N,Y):
        term1 = (((eizz-eizw)/eizw)*((eizz-eizw)/eizw))
        term2 = (Y * eizw *eizw * (coeff*N)**4 * L**4 / (c**4))
        #print 'As term1 = ', term1
        #print 'As term2 = ', term2
        ##print 'As term3 = ', term3
        #print '----'
        return  term1 *  term2# * term3

def A_2s(eizz,eizw, L , N ,Y):
        term1 = (((eizz-eizw)/eizw)*((eizz-eizw)/eizw))
        term2 = (Y * eizw *eizw * (coeff*N)**4 * L**4 / (c**4))
        #print 'A_2s term1 = ', term1
        #print 'A_2s term2 = ', term2
        ##print 'A_2s term3 = ', term3
        #print '----'
        return (term1 * term2)# * term3


y = np.zeros(shape=(len(ns),len(ls)))
y_2 = np.zeros(shape=(len(ns),len(ls)))
A   = np.zeros(shape=(len(ns),len(ls)))
A_2 = np.zeros(shape=(len(ns),len(ls)))
EL = np.zeros(len(ls))
G_l_t_dt = np.zeros(shape=(len(ls),len(thetas)))

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


```

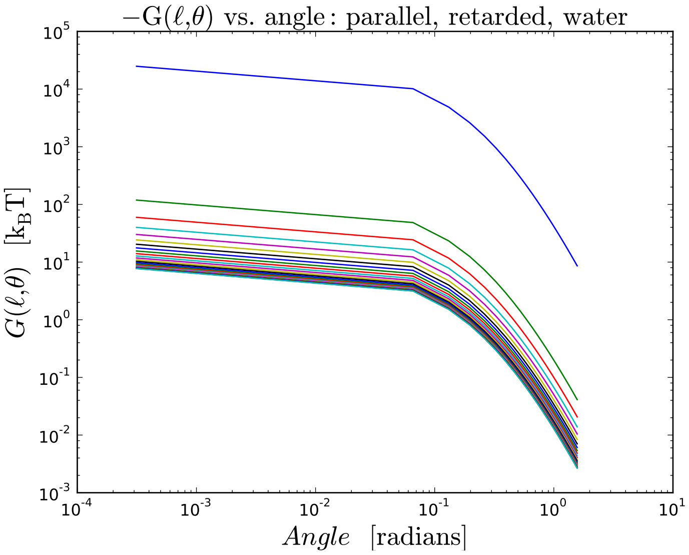
```
for k,length in enumerate(ls):
        sum_A = np.empty(len(ls))
        sum_A_2 = np.empty(len(ls))
        for j,n in enumerate(ns):
                # Integral:
                y[j,k]    = trapz(ys(aiz[j],ts,eiz_w[j],length,n),ts,dt)
                y_2[j,k] = trapz(y_2s(aiz[j],ts,eiz_w[j],length,n),ts,dt)
                #print 'dt Integral   y = ',i,k,j, y
                #print 'dt Integral y_2 = ',i,k,j, y_2
                #print '----'
                #print 'N terms for A0 = '  , As(eiz_z[j],eiz_w[j],length,n,y)
                #print 'N terms for A2 = ', A_2s(eiz_z[j],eiz_w[j],length,n,y_2)
                #print '----'
                A[j,k]    = As(eiz_z[j],eiz_w[j],length,n,y[j,k])
                A_2[j,k] = A_2s(eiz_z[j],eiz_w[j],length,n,y_2[j,k])# * np.cos(2.0*theta)

        sum_A = np.sum(A,axis=0)
        #print 'sum of A0 = ', k,j,sum_A
        sum_A_2 = np.sum(A_2,axis=0)
        #print 'sum of A2 = ', k,j,sum_A_2
        #print '----'
        #print 'shape sum_A_2 = ', np.shape(sum_A_2)
        #sys.exit()
for k,length in enumerate(ls):
        for i, theta in enumerate(thetas):
                EL[k] = 1./(length*length*length*length)
                G_l_t_dt[k,i] = (1.602e-19 / 4.11e-21) * (1./32) * EL[k]*np.pi*r_1*r_1*r_2*r_2*(sum_A[k] + sum_A_2[k]* np.cos(2.0*theta) )/(2.0*np.sin(theta))#  (1e21)*


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'$\varepsilon_{vac}(i\zeta_{N})$')
pl.plot(ns,eiz_w, color = 'c', label = r'$\varepsilon_{water}(i\zeta_{N})$')
pl.xlabel(r'$N$', size = 20)
pl.ylabel(r'$\varepsilon(i\zeta)$', size = 20)
pl.legend(loc = 'best')
pl.title(r'$\mathrm{CG-10\, DNA}$', size = 20)
pl.axis([0,500,0.9,2.6])
pl.savefig('plots/par_ret_water/eiz.pdf' )
pl.show()
show()
```

```

pl.figure()
pl.loglog(ls,(kb*T/32)*sum_A,'b-', label = r'$\mathcal{A^{(0)}}$')
pl.loglog(ls,(kb*T/32)*sum_A_2,'b--', label = r'$\mathcal{A^{(2)}}$')
pl.xlabel(r'$\mathrm{separation}\,\ell\,\,\,\rm{[m]}$', size = 20)
pl.ylabel(r'$\mathrm{\mathcal{-A^{(0)}},\,\,-A^{(2)}}}$', size = 20)
pl.title(r'$\mathrm{Hamaker \, coeff.s \,:\,parallel,\,retarded,\,water}$', size
    = 20)
pl.legend(loc = 'lower right')
pl.axis([1e-9,1e-8,1e-24,1e-18])
pl.savefig('plots/par_ret_water/A0_A2.pdf')
show()
```

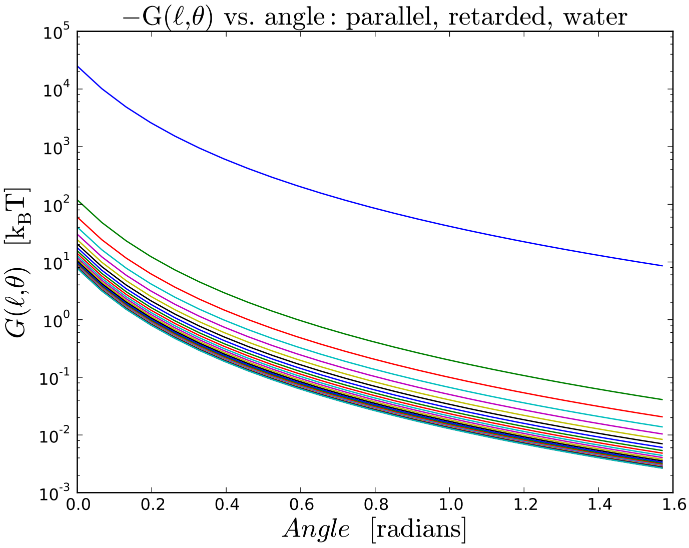


```

pl.figure()
pl.loglog(thetas, G_l_t_dt)#, label = labels_l[k])

pl.xlabel(r'$Angle\,\,\mathrm{[radians]}$', size = 20)
pl.ylabel(r'$G(\ell,\theta)\,\,\mathrm{[k_{B}T]}$', size = 20)
#pl.axis([(1./25)*np.pi,(3./4)*np.pi,105,135])
pl.title(r'$\mathrm{-G(\ell,\theta)\,vs.\,angle:\,parallel,\,retarded,\,water}$'
    , size = 20)
#pl.legend(loc = 'best')
```

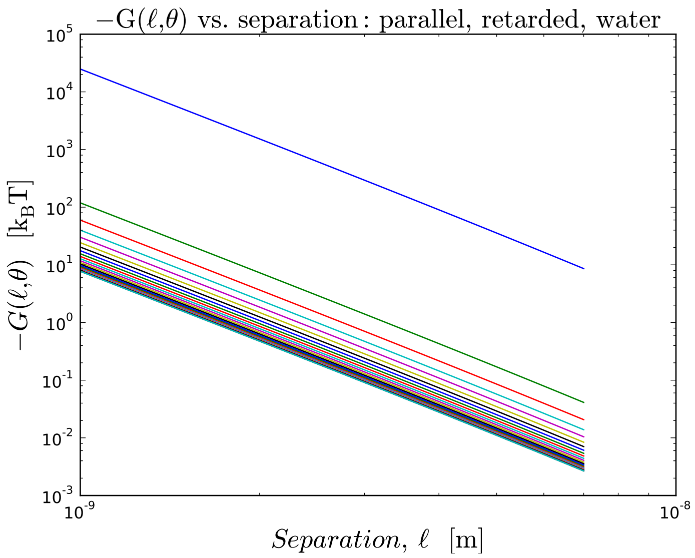
```
pl.savefig('plots/par_ret_water/G_vs_theta.pdf')
show()
```



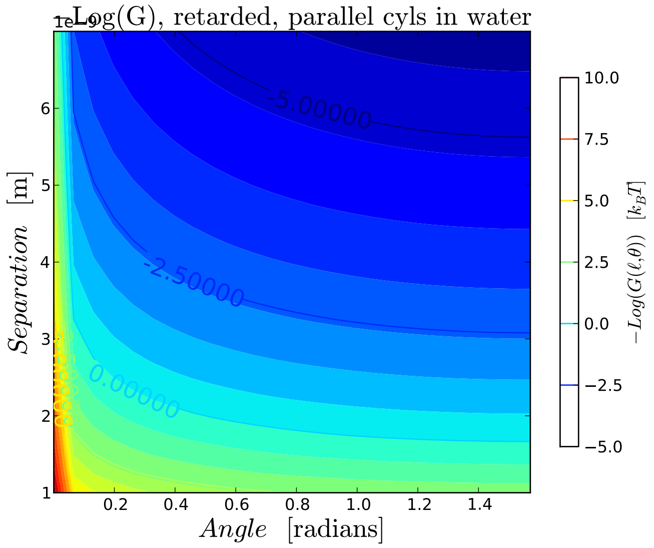
```

pl.figure()
#pl.loglog(thetas, G_l_t_dt)#, label = labels_l[k])
pl.semilogy(thetas, G_l_t_dt)
pl.xlabel(r'$Angle\,\,\mathrm{[radians]}$', size = 20)
pl.ylabel(r'$G(\ell,\theta)\,\,\mathrm{[k_{B}T]}$', size = 20)
#pl.axis([(1./25)*np.pi,(3./4)*np.pi,105,135])
pl.title(r'$\mathrm{-G(\ell,\theta)\,vs.\,angle:\,parallel,\,retarded,\,water}$'
    , size = 20)
#pl.legend(loc = 'best')
pl.savefig('plots/par_ret_water/semilog_G_vs_theta.pdf')
show()
```



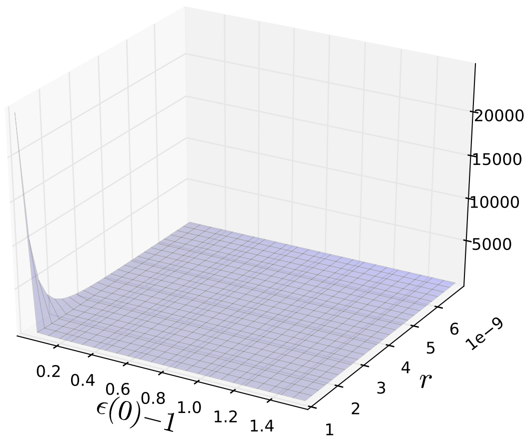
```

pl.figure()
pl.loglog(ls, G_l_t_dt)#, label = labels[i])

pl.xlabel(r'$Separation,\,\ell\,\,\mathrm{[m]}$', size = 20)
pl.ylabel(r'$-G(\ell,\theta)\,\,\mathrm{[k_{B}T]}$', size = 20)
#pl.axis([1.5e-9, 6.5e-8,100,145])
pl.title(r'$\mathrm{-G(\ell,\theta)\,vs.\,separation:\,parallel,\,retarded,\,
    water}$', size = 20)
#pl.legend(loc = 'best')
pl.savefig('plots/par_ret_water/G_vs_l.pdf')
show()
```



```

#G_l_t_dt[G_l_t_dt>300]= 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, np.log(G_l_t_dt), 25)#, cmap = cm.hot())

CS = pl.contour(X,Y,np.log(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'$Angle\,\,\mathrm{[radians]}$', size = 20)
pl.ylabel(r'$Separation\,\,\mathrm{[m]}$', size = 20)
pl.title(r'$\mathrm{-Log(G),\,retarded,\,parallel\,cyls\,in\,water}$', size =
    20)#uas a function of separation and angle')

cbar = pl.colorbar(CS, shrink = 0.8, extend = 'both')
cbar.ax.set_ylabel(r'$-Log(G(\mathcal{\ell},\theta))\,\,[k_{B}T]$', size = 14)
cbar.add_lines(CS)
##pl.axis([0,1.0,0,1.0])
#pl.grid()
pl.savefig('plots/par_ret_water/logG_contour.pdf')
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()
```


20000
15000
10000
5000
0.2
0.4
0.6
0.8
1.0
1.2
1.4
$\epsilon(0)-1$
1
2
3
4
5
6
1e−9
r

```
#pp.savefig()
```