Numerically solve: Equation 27: $g(\ell) = -\frac{k_B T}{32}\frac{R_1^2 R_2^2}{\ell^5}\sum_{n=0}^{\infty}{}'\Delta_{1,\parallel}\Delta_{2,\parallel}\int_1^{+\infty}\frac{dy}{\sqrt{y^2-1}}\int_0^{\infty}udu\,\frac{e^{-2y\sqrt{u^2+p_n^2}}}{(u^2+p_n^2)^{1/2}}\,h(a_1(i\omega_n),a_2(i\omega_n),u,p_n^2),and h(a_1(i\omega_n$
$2\left[(1+3a_1)(1+3a_2)u^4+2(1+2a_1+2a_2+3a_1a_2)u^2p_n^2+2(1+a_1)(1+a_2)p_n^4\right]+$
$(1-a_1)(1-a_2)(u^2+2p_n^2)^2$.

/usr/bin/python

```
import numpy as np
import scipy.optimize as opt
from scipy.integrate import trapz
import matplotlib.pyplot as pl
import pyreport
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
```

Problem 1) Prove that: $= \frac{1}{N-1}\left(N\sigma^2 - \sigma^2\right)$

Null Hypothesis: the radioactive counts have a Poisson distribution with mean μ.

```

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_x = np.loadtxt('data/eiz_x_output.txt') #perpendicular, radial
#eiz_y = np.loadtxt('data/eiz_y_output.txt')
#eiz_z = np.loadtxt('data/eiz_z_output.txt') # parallel,axial
#eiz_w = np.loadtxt('data/eiz_w_output.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
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(1.0,501.0)
z = ns * coeff
```

```
ls = np.linspace(0.1e-9, 7.0e-8, 20)
#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):
```

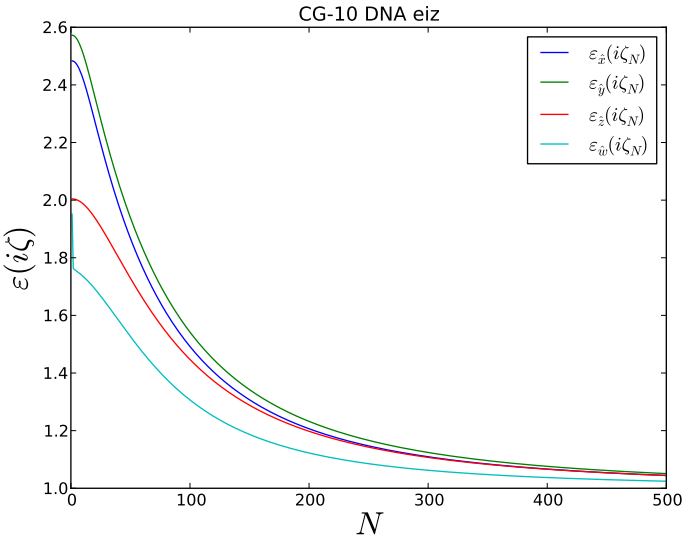
```
124 #       print "on n=%d of %d"%(j,len(ns))
125 #       for k,l in enumerate(ls):
126 #               # Integrand:
127 #               y_arg   = ys(aiz[j],ts,eiz_w[j],l,n)
128 #               y_2_arg = y_2s(aiz[j],ts,eiz_w[j],l,n)
129 #               # Integral:
130 #               y   = trapz(y_arg,ts,dt)
131 #               y_2 = trapz(y_2_arg,ts,dt)
132 #               A[j,k]   = As(eiz_z[j],eiz_w[j],l,n,y)
133 #               for i, theta in enumerate(thetas):
134 #                       A_2[j,k,i] = A_2s(eiz_z[j],eiz_w[j],l,n,y_2) * np.cos
        (2.0*theta)
135 #sum_A = np.sum(A,axis=0)
136 #sum_A_2 = np.sum(A_2,axis=0)
137 pl.figure
138 for i, theta in enumerate(thetas):
139         print 'i,theta = ', (i,theta)
140         for k,length in enumerate(ls):
141                 for j,n in enumerate(ns):
142                         #print "on n=%d of %d"%(j,len(ns))
143                         # Integrand:
144                         y_arg   = ys(aiz[j],ts,eiz_w[j],length,n)
145                         y_2_arg = y_2s(aiz[j],ts,eiz_w[j],length,n)
146                         # Integral:
147                         y   = trapz(y_arg,ts,dt)
148                         y_2 = trapz(y_2_arg,ts,dt)
149                         A[j,k]   = As(eiz_z[j],eiz_w[j],length,n,y)
150                         A_2[j,i,k] = A_2s(eiz_z[j],eiz_w[j],length,n,y_2) * np.
                            cos(2.0*theta)
151                 sum_A = np.sum(A,axis=0)
152                 #print 'shape sum_A = ', np.shape(sum_A)
153                 sum_A_2 = np.sum(A_2,axis=0)
154                 #print 'shape sum_A_2 = ', np.shape(sum_A_2)
155                 #sys.exit()
156                 EL[k] = 1./(length*length*length*length)
157                 G_l_t_dt[i,k] = 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)*
158
159                 labels = 'theta = %.1f' %(theta)
160                 #print 'theta = %.1f, length = %.1f, G = %s' %(i,k,G_l_t_dt[k,i]
                    )
161         pl.plot(ls, G_l_t_dt, label = labels[i])
```

```
i,theta =  (0, 0.00031415926535897931)
```

Error:

```
Traceback (most recent call last):
  File "/usr/local/lib/python2.7/dist-packages/pyreport-0.3.4c-py2.7.egg/pyreport/main.py"
    exec block_text in self.namespace
  File "<string>", line 26, in <module>
  File "/usr/lib/pymodules/python2.7/matplotlib/pyplot.py", line 2467, in plot
    ret = ax.plot(*args, **kwargs)
  File "/usr/lib/pymodules/python2.7/matplotlib/axes.py", line 3893, in plot
    for line in self._get_lines(*args, **kwargs):
  File "/usr/lib/pymodules/python2.7/matplotlib/axes.py", line 322, in _grab_next_args
    for seg in self._plot_args(remaining, kwargs):
  File "/usr/lib/pymodules/python2.7/matplotlib/axes.py", line 300, in _plot_args
    x, y = self._xy_from_xy(x, y)
  File "/usr/lib/pymodules/python2.7/matplotlib/axes.py", line 240, in _xy_from_xy
    raise ValueError("x and y must have same first dimension")
ValueError: x and y must have same first dimension
```

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



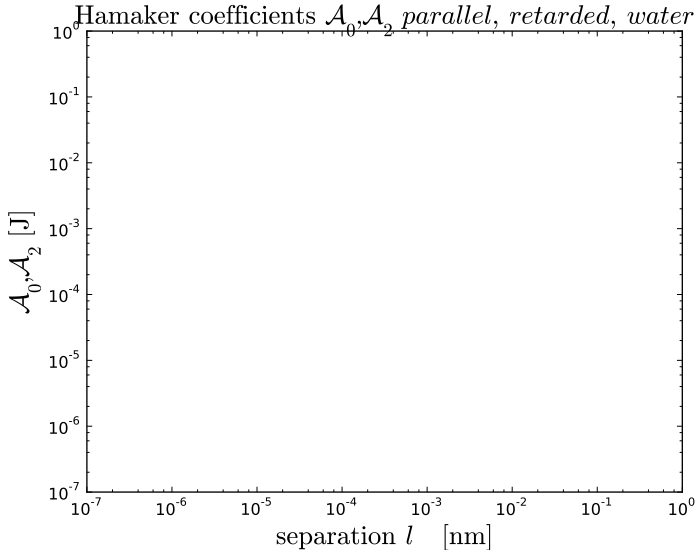

```
159
160 pl.figure()
161 pl.loglog(ls,sum_A,'b-',ls,sum_A_2,'b--')
```

Error:

```
Traceback (most recent call last):
  File "/usr/local/lib/python2.7/dist-packages/pyreport-0.3.4c-py2.7.egg/pyreport/main.py"
    exec block_text in self.namespace
  File "<string>", line 3, in <module>
  File "/usr/lib/pymodules/python2.7/matplotlib/pyplot.py", line 2395, in loglog
    ret = ax.loglog(*args, **kwargs)
  File "/usr/lib/pymodules/python2.7/matplotlib/axes.py", line 4026, in loglog
    l = self.plot(*args, **kwargs)
  File "/usr/lib/pymodules/python2.7/matplotlib/axes.py", line 3893, in plot
    for line in self._get_lines(*args, **kwargs):
  File "/usr/lib/pymodules/python2.7/matplotlib/axes.py", line 322, in _grab_next_args
    for seg in self._plot_args(remaining, kwargs):
  File "/usr/lib/pymodules/python2.7/matplotlib/axes.py", line 300, in _plot_args
    x, y = self._xy_from_xy(x, y)
  File "/usr/lib/pymodules/python2.7/matplotlib/axes.py", line 240, in _xy_from_xy
    raise ValueError("x and y must have same first dimension")
ValueError: x and y must have same first dimension
```

```
161 pl.xlabel(r'$\mathrm{separation}\,\it{l}\,\,\,\rm{[nm]}$', size = 20)
162 pl.ylabel(r'$\mathrm{\mathcal{A_{0},A_{2}}\,[J]}$', size = 20)
163 pl.title(r'$\mathrm{Hamaker \, coefficients \,\mathcal{A_{0},A_{2}}}\, parallel
    ,\,retarded,\,water$', size = 20)
```

```
164 show()
```




```
165
166 #for i, theta in enumerate(thetas):
167 #    for h, length in enumerate(ls):
168 #        EL[h] = 1./(length*length*length*length)
169 #        #G_l_t_dt[i,h] = -np.log(EL[h]*np.pi*r_1*r_1*r_2*r_2*(sum_A[h] +
    sum_A_2[h] * np.cos(2.0*theta))/(2.0*np.sin(theta)))# NOTE: added in -log*
    for plotting purposes
170 #        #G_l_t_dt[i,h] = kb * T * (1./32) * EL[h]*np.pi*r_1*r_1*r_2*r_2*(sum_A[
    h] + sum_A_2[h] * np.cos(2.0*theta))/(2.0*np.sin(theta))
171 #        G_l_t_dt[i,h] = kb * T * (1./32) * EL[h]*np.pi*r_1*r_1*r_2*r_2*(sum_A[h
    ] + sum_A_2[h])/(2.0*np.sin(theta))
172 #        print 'theta = %.1f, length = %.1f, G = %s' %(i,h,G_l_t_dt[i,h])
173
174 # CONTOUR PLOT:
175 X,Y = np.meshgrid(thetas, ls)
176 pl.figure()
177 pl.contourf(X, Y, G_l_t_dt, 10)#, cmap = cm.hot())
```

Error:

```
Traceback (most recent call last):
  File "/usr/local/lib/python2.7/dist-packages/pyreport-0.3.4c-py2.7.egg/pyreport/main.py"
    exec block_text in self.namespace
  File "<string>", line 3, in <module>
  File "/usr/lib/pymodules/python2.7/matplotlib/pyplot.py", line 2215, in contourf
    ret = ax.contourf(*args, **kwargs)
  File "/usr/lib/pymodules/python2.7/matplotlib/axes.py", line 7387, in contourf
    return mcontour.QuadContourSet(self, *args, **kwargs)
  File "/usr/lib/pymodules/python2.7/matplotlib/contour.py", line 1112, in __init__
    ContourSet.__init__(self, ax, *args, **kwargs)
  File "/usr/lib/pymodules/python2.7/matplotlib/contour.py", line 703, in __init__
    self._process_args(*args, **kwargs)
  File "/usr/lib/pymodules/python2.7/matplotlib/contour.py", line 1125, in _process_args
    x, y, z = self._contour_args(args, kwargs)
  File "/usr/lib/pymodules/python2.7/matplotlib/contour.py", line 1172, in _contour_args
    x,y,z = self._check_xyz(args[:3], kwargs)
  File "/usr/lib/pymodules/python2.7/matplotlib/contour.py", line 1209, in _check_xyz
    raise TypeError("Inputs x and y must be 1D or 2D.")
TypeError: Inputs x and y must be 1D or 2D.
```

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

Error:

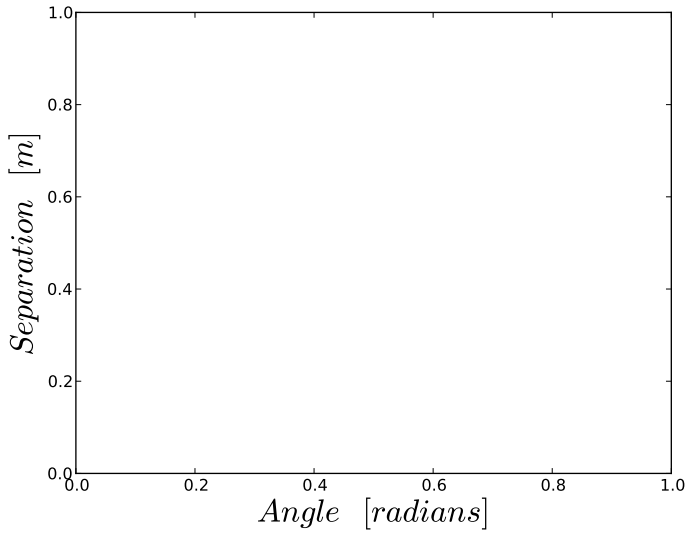
```
Traceback (most recent call last):
  File "/usr/local/lib/python2.7/dist-packages/pyreport-0.3.4c-py2.7.egg/pyreport/main.py"
    exec block_text in self.namespace
  File "<string>", line 3, in <module>
  File "/usr/lib/pymodules/python2.7/matplotlib/pyplot.py", line 2197, in contour
    ret = ax.contour(*args, **kwargs)
  File "/usr/lib/pymodules/python2.7/matplotlib/axes.py", line 7381, in contour
    return mcontour.QuadContourSet(self, *args, **kwargs)
  File "/usr/lib/pymodules/python2.7/matplotlib/contour.py", line 1112, in __init__
    ContourSet.__init__(self, ax, *args, **kwargs)
  File "/usr/lib/pymodules/python2.7/matplotlib/contour.py", line 703, in __init__
    self._process_args(*args, **kwargs)
  File "/usr/lib/pymodules/python2.7/matplotlib/contour.py", line 1125, in _process_args
    x, y, z = self._contour_args(args, kwargs)
  File "/usr/lib/pymodules/python2.7/matplotlib/contour.py", line 1172, in _contour_args
    x,y,z = self._check_xyz(args[:3], kwargs)
  File "/usr/lib/pymodules/python2.7/matplotlib/contour.py", line 1209, in _check_xyz
    raise TypeError("Inputs x and y must be 1D or 2D.")
TypeError: Inputs x and y must be 1D or 2D.
```

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

Error:

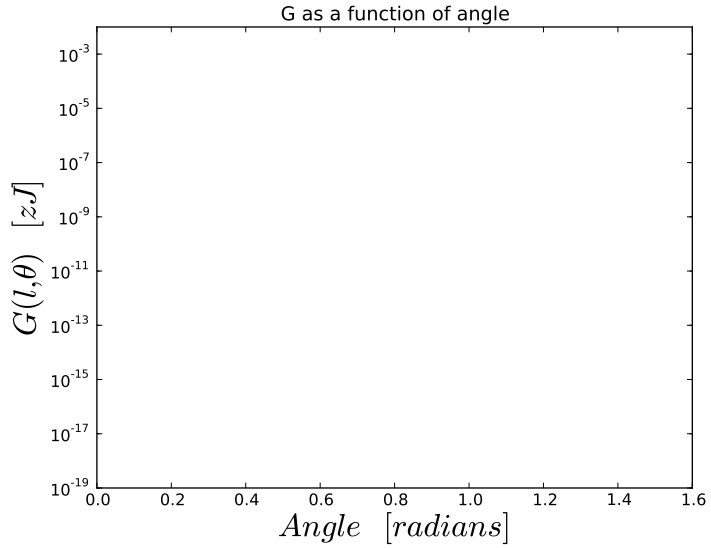
```
Traceback (most recent call last):
  File "/usr/local/lib/python2.7/dist-packages/pyreport-0.3.4c-py2.7.egg/pyreport/main.py"
    exec block_text in self.namespace
  File "<string>", line 3, in <module>
NameError: name 'CS' is not defined
```

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




```
189
190 fig = pl.figure()
191 ax = fig.gca(projection = '3d')
192 #ax.text(-7, 6, 0.7, r'$\zeta/\omega_{0}$', zdir = (-1,1,-3), size = 21)
193 pl.figure()
194 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)
195
196 #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()
197 #colorbar(surf)
198 #cbar.ax.set_ylabel(r'$\frac{\xi}{\omega_{0}}$', size = 24)
199 #cset = ax.contour(X,Y,h, zdir = 'z', offset = 0, cmap = cm.jet)
200 #cset = ax.contour(X,Y,h, zdir = 'x', offset = 5, cmap = cm.jet)
201 #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
202 #ax.view_init(elev = 19, azim = -112)
203 #zlabel(r'$\xi/\omega_{0}$', size = 21)
204 #ylabel(r'$r$', size = 24)
205 #xlabel(r'$(\epsilon(0) -1)$', size = 24)
206 #text = Axes.text(self, x, y, s, **kwargs)
207 #art3d.text_2d_to_3d(text, z, zdir)
208 #return text
209 #pl.text(6,0, 0, r'$\xi/\omega_{0}$',size = 21 ,rotation = 'horizontal')
210 #ax.text(r'$\xi/\omega_{0}$',6,0, 0, size = 21 ,rotation = 'horizontal')
211 #ax.set_zlabel(r'$\xi/\omega_{0}$',size = 21 ,rotation = 'horizontal' )
212 ax.set_xlabel(r'$\epsilon(0)-1$', size = 21)
213 ax.set_ylabel(r'$r$', size = 22)
214 show()
```

```
214
215 pl.figure()
216 pl.semilogy(thetas, G_l_t_dt)
217 pl.xlabel(r'$Angle\,\,[radians]$', size = 24)
218 pl.ylabel(r'$G(l,\theta)\,\,[zJ]$', size = 24)
219 #pl.axis([(1./25)*np.pi,(3./4)*np.pi,105,135])
220 pl.title('G as a function of angle')
221 show()
```




```
222
223 pl.figure()
224 pl.loglog(ls, G_l_t_dt)
```

Error:

```
Traceback (most recent call last):
  File "/usr/local/lib/python2.7/dist-packages/pyreport-0.3.4c-py2.7.egg/pyreport/main.py"
    exec block_text in self.namespace
  File "<string>", line 3, in <module>
  File "/usr/lib/pymodules/python2.7/matplotlib/pyplot.py", line 2395, in loglog
    ret = ax.loglog(*args, **kwargs)
  File "/usr/lib/pymodules/python2.7/matplotlib/axes.py", line 4026, in loglog
    l = self.plot(*args, **kwargs)
  File "/usr/lib/pymodules/python2.7/matplotlib/axes.py", line 3893, in plot
    for line in self._get_lines(*args, **kwargs):
```

```
  File "/usr/lib/pymodules/python2.7/matplotlib/axes.py", line 322, in _grab_next_args
    for seg in self._plot_args(remaining, kwargs):
  File "/usr/lib/pymodules/python2.7/matplotlib/axes.py", line 300, in _plot_args
    x, y = self._xy_from_xy(x, y)
  File "/usr/lib/pymodules/python2.7/matplotlib/axes.py", line 240, in _xy_from_xy
    raise ValueError("x and y must have same first dimension")
ValueError: x and y must have same first dimension
```

```
224 pl.xlabel(r'$Separation\,\,[m]$', size = 24)
225 pl.ylabel(r'$G(l,\theta)\,\,[zJ]$', size = 24)
226 #pl.axis([1.5e-9, 6.5e-8,100,145])
227 pl.title('G as a function of separation')
228 show()
```