Final Exam

December 5, 2015

CBE 521, Fall 2014

1 Brandon Lampe

2 Mean Squared Displacement

The mean square displacement $\langle x^2 \rangle$ of a Brownian particle in one dimension is defined by Einstein as :

$$\langle x^2 \rangle = \int_{-\infty}^{\infty} x^2 f(x,t) dx$$

2.1 (a) Find $\langle x^2 \rangle$ if:

$$f(x,t) = \frac{e^{-\left(\frac{x^2}{4Dt}\right)}}{\sqrt{4\pi Dt}}$$

then $\langle x^2 \rangle$ is:

$$\langle x^2 \rangle = \frac{1}{\sqrt{4\pi Dt}} \int_{-\infty}^{\infty} x^2 e^{-\left(\frac{x^2}{4Dt}\right)} dx$$

```
In [1]: from sympy import *
        from sympy import exp, sqrt, pi, oo
        init_printing()
        from IPython.display import display

        #Plotting
        %matplotlib inline
        import matplotlib.pyplot as plt
```

```
In [2]: x = Symbol('x', real=True, positive=True) # position
        D = Symbol('D', real=True, positive=True) # diffusion coef.
        t = Symbol('t', real=True, positive=True) # time

        gaussian_2 = x**2 * exp(-(x**2)/(4. * D * t))/sqrt(4 * pi * D * t)
        integrate(gaussian_2, (x,-oo,oo), conds = 'none')
```

Out[2]:

$$2.0Dt$$

A Gaussian function describes the the position (random walk) of a particle, and by integrating over all possible locations, the intuitive result of 2Dt is obtained.

2.2 (b) Show that the mean displacement is:

$$\langle x \rangle = \int_{-\infty}^{\infty} x f(x,t) dx = 0$$

Let

$$a = \frac{1}{\sqrt{\pi D t}} \quad (1)$$
$$u = -\frac{x^2}{4Dt} \quad (2)$$
$$then \quad (3)$$
$$\langle x \rangle = a \int_{-\infty}^{\infty} x e^u du = a\left[(u-1)e^u\right] \quad (4)$$

Which when evaluated at the integration limits of $x = \infty$ and $x = -\infty$ results in $\langle x \rangle = 0$

In [3]:
```
gaussian_1 = x * exp(-(x**2)/(4. * D * t))/sqrt(4 * pi * D * t)
integrate(gaussian_1, (x,-oo,oo), conds = 'none')
```

Out[3]:

$$0$$

This result is also a very intuitive.

2.3 (c) Find particle displacement in infinite half space

The mean displacement $\langle x \rangle$ of a Brownian particle in one dimension in infinite half space is described by:

$$\langle x \rangle = \int_{0}^{\infty} x f(x,t) dx$$

In [4]:
```
integrate(gaussian_2, (x,0,oo), conds = 'none')
```

Out[4]:

$$1.0Dt$$

A Gaussian function describes the the position (random walk) of a particle, and by integrating over the infinite half space, the result of 1Dt is obtained.

3 Dynamic Light Scattering of Mono Dispersed Sample

A measurement of the time correlation function by Dynamic Light Scattering gave the data summarized in the table on the right. Find the effective diffusion coefficient if the wave number is q = 1.87x107. Assume that the sample is mono dispersed.

t	g^1(t)
0.000	1.000
0.001	0.498
0.002	0.248
0.003	0.124
0.004	0.061
0.005	0.030

t	g^1(t)
0.006	0.015
0.007	0.008
0.008	0.004
0.009	0.002
0.010	0.001

Data will be fit to the following equation:

$$g^1(q,t) = exp\left(-q^2\bar{D}t\right)$$

Where $\bar{D}$ is the effective diffusion coeficient.

3.0.1 Write numerical routine to fit data

```
In [5]: import numpy as np
        import scipy.optimize
```

Write function to be minimized (objective function) and function for viewing the results

```
In [19]: def objective_func(D_guess, q, t_meas, g_meas):
             Tau = q**2 * D_guess # decay rate
             g_calc = np.exp(-Tau*t_meas)
             error = (np.log(g_calc) - np.log(g_meas))# linearize the error by taking log of exponentia
             error_norm = np.linalg.norm(error)
             return error_norm

         def calc_ln_g(D, q, t): # calculates the log of the accumulation function
             Tau = q**2 * D # decay rate
             ln_g = -Tau*t
             return ln_g
```

The Measured Data

```
In [20]: t_meas = np.linspace(0, 0.01,11) #time
         g_meas = np.array([1.000,0.498,0.248,0.124,0.061,0.030,0.015,
                            0.008,0.004,0.002,0.001], dtype = np.double) #measured autocorrelation data
```

Run optimization routine

```
In [21]: ln_g_meas = np.log(g_meas) # take ln of accumulation function
         q = 1.87*10**7 # wave vector
         D_guess = 10.0**-12 # initial guess

         res = scipy.optimize.minimize(objective_func, D_guess, args=(q, t_meas, g_meas), method='Nelde
                                       options={'xtol': 1e-8, 'disp': True})
         print(res)
```

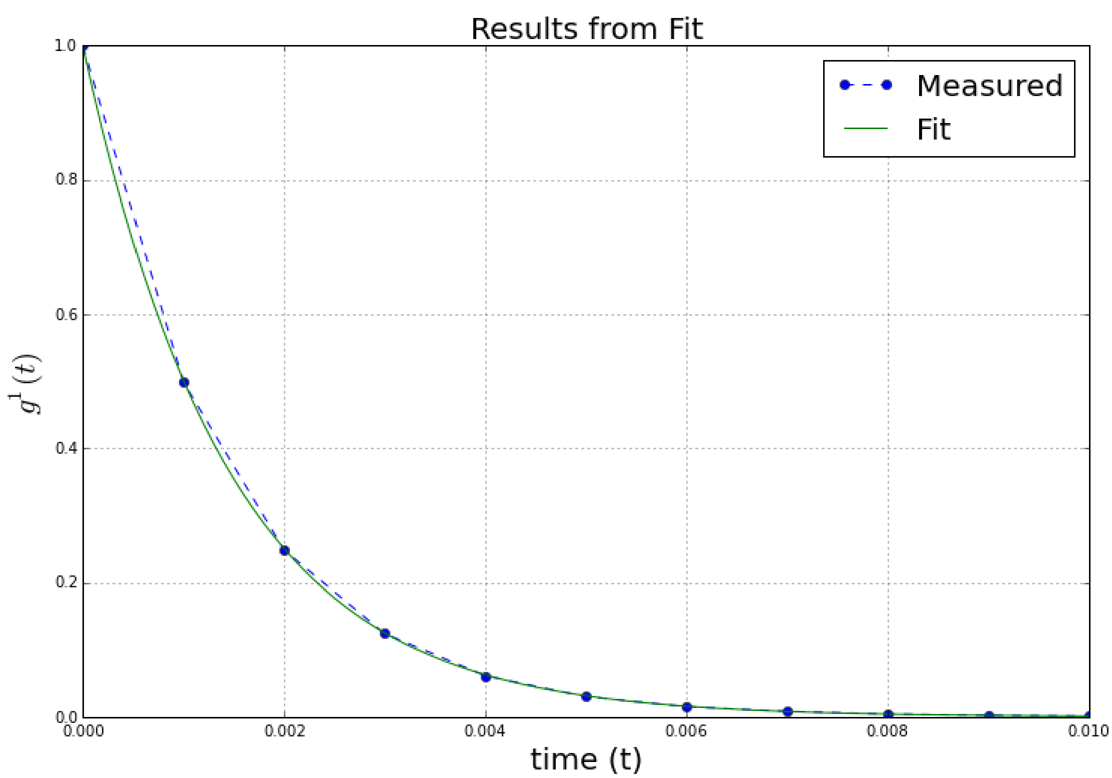
```
Optimization terminated successfully.
         Current function value: 0.078857
         Iterations: 13
         Function evaluations: 26
  status: 0
    nfev: 26
 success: True
```

```
     fun: 0.0788573415956111197
       x: array([  1.98125000e-12])
 message: 'Optimization terminated successfully.'
     nit: 13
```

** Plot Results of Fit**

In [22]:
```python
D_calc = res.x[0]
print(D_calc)

# the calculated auto correlation function
t_plot = np.linspace(0,0.01,101)
g_calc = np.exp(calc_ln_g(D_calc, q, t_plot))

fig_2, ax = plt.subplots(figsize = (12,8))
ax.plot(t_meas, g_meas,'o--', lw=1)
ax.plot(t_plot, g_calc, lw=1)
ax.legend(('Measured', 'Fit'),framealpha = 1, loc=0, fontsize=20)

ax.set_xlabel('time (t)', fontsize = 20)
ax.set_ylabel(''r'$g^{1}(t)$', fontsize = 20)
ax.set_title('Results from Fit' , fontsize = 20)
ax.grid(b = True, which = 'major')
ax.grid(b = True, which = 'major')

fig_name = 'prob_2.pdf'
path = '/Users/Lampe/Documents/UNM_Courses/CBE-521/Exam02/'
fig_2.savefig(path + fig_name)
```

```
1.98125e-12
```



The fit to data resulted in a calculed effective diffusion coefficient of:

$$\bar{D} = 1.981 \times 10^{-12}$$

4 Dynamic Light Scattering of Bidispersed Sample

A measurement of the time correlation function by Dynamic Light Scattering gave the data summarized in the table on the right. Find the effective diffusion coefficient if the wave number is q = 1.87x107. Assume that the sample is bidispersed.

t	g^1(t)
0.000	1.000
0.001	0.373
0.002	0.155
0.003	0.069
0.004	0.033
0.005	0.016
0.006	0.008
0.007	0.004
0.008	0.002
0.009	0.001

Data will be fit to the following equation:

$$g^1(q,t) = \frac{1}{2}\left[exp\left(-2q^2\bar{D}_1 t\right) + exp\left(-2q^2\bar{D}_2 t\right)\right]$$

Where $\bar{D}_1$ and $\bar{D}_2$ are the effective diffusion coeficients.

4.0.2 Write a numerical routine to fit the data

Write function to be minimized (objective function) and function for viewing the results

```
In [14]: def objective_func2(D, q, t_meas, g_meas):
             D_guess1 = D[0]
             D_guess2 = D[1]
             Tau1 = q**2 * D_guess1 # decay rate
             Tau2 = q**2 * D_guess2 # decay rate
             g_calc = 0.5* (np.exp(-2*Tau1*t_meas)+np.exp(-2*Tau2*t_meas))
             error = (np.log(g_calc) - np.log(g_meas))# linearize the error by taking log of exponentia
             error_norm = np.linalg.norm(error)
             return error_norm

         def calc_ln_g2(D_1, D_2, q, t): # calculates the log of the accumulation function
             Tau1 = q**2 * D_1 # decay rate
             Tau2 = q**2 * D_2 # decay rate

             ln_2g = -Tau1*t -Tau2*t
             return ln_2g
```

The Measured Data

```
In [15]: t_meas = np.linspace(0, 0.009,10) #time
         g_meas = np.array([1.000,0.373,0.155,0.069,0.033, 0.016,
                            0.008,0.004,0.002,0.001], dtype = np.double) #measured autocorrelation data
```

Run optimization routine

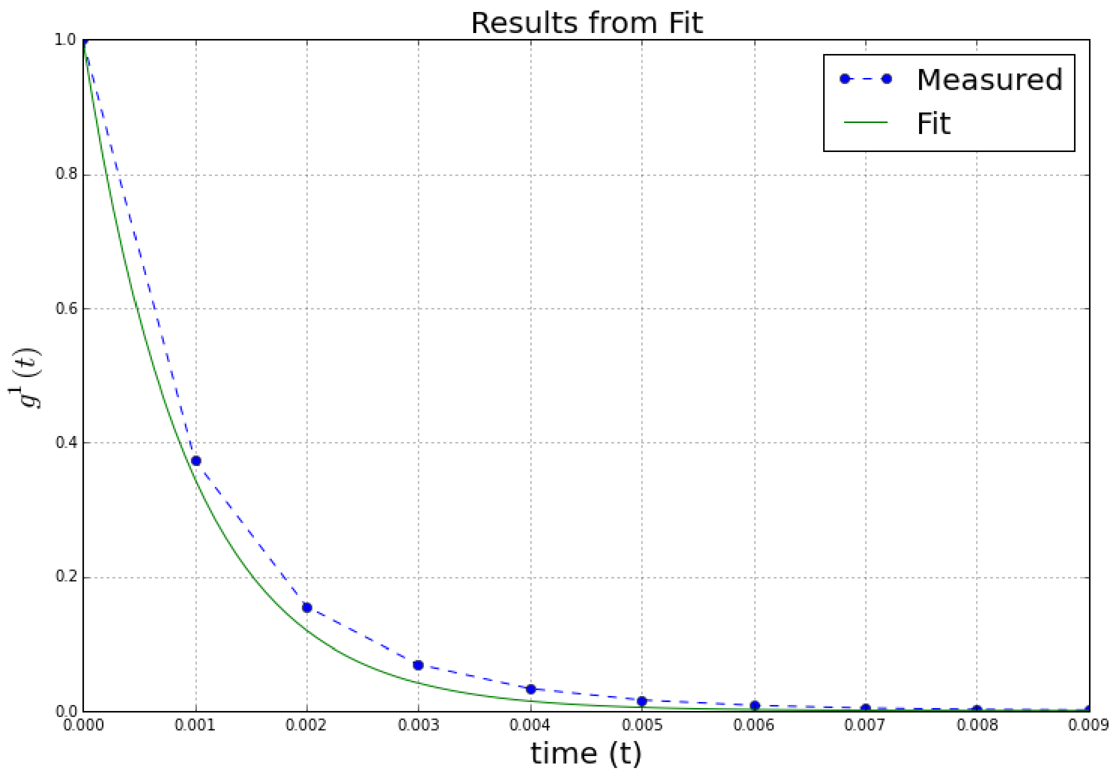
```
In [16]: ln_g_meas = np.log(g_meas) # take ln of accumulation function
         q = 1.87*10**7 # wave vector
         D_guess1 = 10.0**-12 # initial guess
         D_guess2 = 10.0**-12 # initial guess

         res = scipy.optimize.minimize(objective_func2, [D_guess1, D_guess2], args=(q, t_meas, g_meas),
                                       options={'xtol': 1e-8, 'disp': True})
         print(res)
```

```
Optimization terminated successfully.
         Current function value: 0.019864
         Iterations: 37
         Function evaluations: 69
  status: 0
    nfev: 69
 success: True
     fun: 0.019863615028497702
       x: array([  2.04940567e-12,   9.88102622e-13])
 message: 'Optimization terminated successfully.'
     nit: 37
```

** Plot Results of Fit**

```
In [18]: D_calc1 = res.x[0]
         D_calc2 = res.x[1]
         print(res.x)

         # the calculated auto correlation function
         t_plot = np.linspace(0,0.009,100)
         g_calc = np.exp(calc_ln_g2(D_calc1,D_calc2, q, t_plot))

         fig_3, ax = plt.subplots(figsize = (12,8))
         ax.plot(t_meas, g_meas,'o--', lw=1)
         ax.plot(t_plot, g_calc, lw=1)
         ax.legend(('Measured', 'Fit'),framealpha = 1, loc=0, fontsize=20)

         ax.set_xlabel('time (t)', fontsize = 20)
         ax.set_ylabel(''r'$g^{1}(t)$', fontsize = 20)
         ax.set_title('Results from Fit' , fontsize = 20)
         ax.grid(b = True, which = 'major')
         ax.grid(b = True, which = 'major')

         fig_name = 'prob_3.pdf'
         path = '/Users/Lampe/Documents/UNM_Courses/CBE-521/Exam02/'
         fig_3.savefig(path + fig_name)
```

```
[  2.04940567e-12   9.88102622e-13]
```



The fit to data resulted in a calculed effective diffusion coefficients of:

$$\bar{D}_1 = 2.049 \times 10^{-12}$$

$$\bar{D}_2 = 9.881 \times 10^{-13}$$