Brownian dynamics and error bar plot

Zhe Chen

July 3, 2017

1 Theoretical Framework

1.1 Brownian Dynamics

Considering hundreds thousand of particles walking in the 2D plane freely without interaction, one can get their probability distribution as Gaussion Distribution as a natural consequence of diffusion equation with gaussian initialization and infinite boundary.

Hence, we can get the probability distribution of these particles as follows, which is a 2D gaussion distribution,

$$P(\vec{r},t) = \frac{1}{2\pi\sigma(t)^2} exp(-\frac{-|\vec{r}|^2}{2\sigma(t)^2}) \tag{1}$$

How to analysis $\sigma(t)$ theoretically

To gain $\sigma(t)$, we need to come to 2D diffusion equation first. Since the brownian motion can be described as diffusion equation: $\frac{\partial\rho}{\partial t} = D\nabla^2\rho$, we can the solution:

$$\rho(r,t) = \frac{1}{4\pi Dt} exp(\frac{-r^2}{4Dt})$$

Hence,

$$\sigma(t) = \sqrt{2Dt} \tag{2}$$

Simulation

In order to simulate the brownian dynamics, we get the equation of particle motion as follows, where D is diffusion coefficient, Δt is time step.

$$r^{n+1} = r^n + \sqrt{2D\Delta t} \cdot \mathcal{N}(0,1)$$

To obtain $\sigma(t)$, we use linear regression as follows, so we can acquire $\sigma(t)$ from its slope.

$$ln(P(\vec{r},t) \sim r^2$$

Attention should to be paid that we must run randomly hundreds or even more times to estimate the deviation of ln(P), since the deviation is decreasing as $N^{-\frac{1}{2}}$, where N is times of running, due to Monte-Carlo theory.

Codes

Please see brownian_dynamics.m and prob_gen.m to get the code implementation, where code commments have been written well.

'brownian_dynamics.m' is to run the simulation and output prob.mat document, which records the probability of r^2 at each repeat running.

'prob_gen' is to gain probability distribution of P(r) according to input of location of all the particles.

1.2 Error bar plot

Because our works are based on statistics, efforts must be made to estimate the deviation of any statistical quantity. As a consequence of Monte-Carlo, we can estimate the error of ln(p) as below.

$$err(ln(P)) = \frac{2Std}{\sqrt{N}}$$

,where N is the number of repeat times and Std is standard deviation (i.e. std function in matlab) of ln(p) over all the repeat runnings.

Codes

Please see 'error_bar_plot' to get the code commments for details. It can make the error-bar figure at each time point and output the image as 't=xx.png'.

2 Numerical Implementation

Some parameters are to be set here.

$N_p = 1024*16$: Particles' number, which is the same with 'brownian_walker' on github.

$dt = 1$: Time step Δt

$t\_steps = 1000$: How many time steps to iterate.

$sample = 100$: Prob Distribution will be recorded at every 'sample' time steps.

$repeat = 500$: The procedure will be ran 'repeat' times.

$r\_step = 10$: r interval to sample on $ln(p) \sim r^2$

$r\_num = 10$: The number of r_step where we calculate the P(r).

$D = 1$: The diffusion coefficient.

How to set the initial distribution of particles.

Since it is gaussion, we should only make sure of one parameter: $\sigma 0$.

Define packing fraction as $\phi = \pi a^2 n_0 \sim 1$,where a is radius of particles and we always set a=1.

Since n satisfies gaussion distribution:

$$n(r, t = 0) = n_0 e^{-r^2/2\sigma_0^2} 2\pi r$$

So: $n_0 = \frac{N}{\int_0^\infty e^{-r^2/2\sigma_0^2} 2\pi r dr} = \frac{N}{2\pi\sigma_0^2}$

Since $\phi \sim 1$, $\sigma 0 = \sqrt{N/2}a \approx 90$

So, we can set $sigma0 = 90$, which is the initial sigma of gaussian distribution.

How to calculate P(r,t)

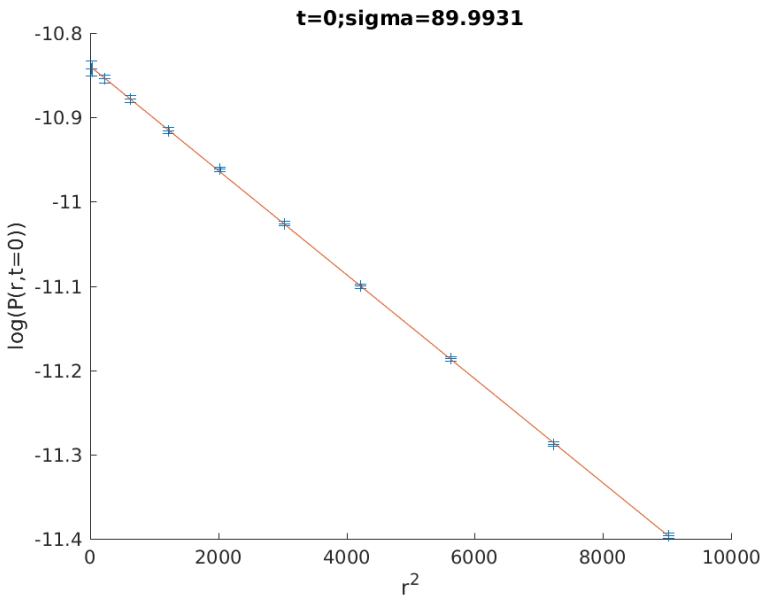
As Eq. 1 shows, the prob of $\vec{r}$ is all the same with the same r, we can count the particles in the annulus whose area is $2\pi r \cdot dr$. So $P(r) = N_{annulus}/(Ndr \cdot 2\pi r)$

3 Results

In this section, we are going to talk about result of the numerical implementation in section 2.

We got 10 error bar figures with t varies from 0 to 900 and we only show 3 figures where t=0,500,900 (i.e. pic 1,2,3 and the rest are saved at './pic/'.

Figure 1: t=0



As we can see, slope is decreasing as t increases. Moreover, the slope is represented by sigma, since Eq. 1 told me that $slope \triangleq k = -\frac{1}{2\sigma(t)^2}$. So, $\sigma = \sqrt{-1/(2k)}$.

So, it's clear that this straight line is getting more and more flat with t increases, since sigma is getting larger and larger.

According to eq. 2, we can compare the σ we got here from simulation with theoretical solution eq. 2.

Figure 2: t=500

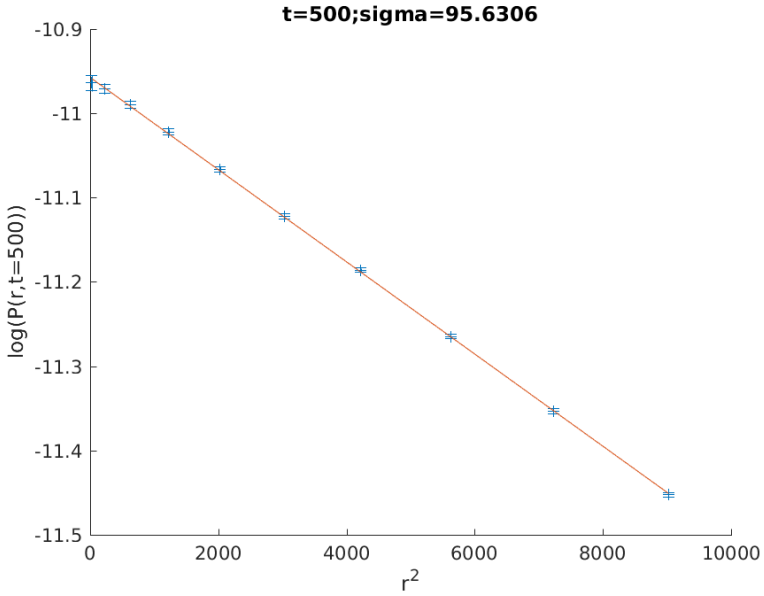
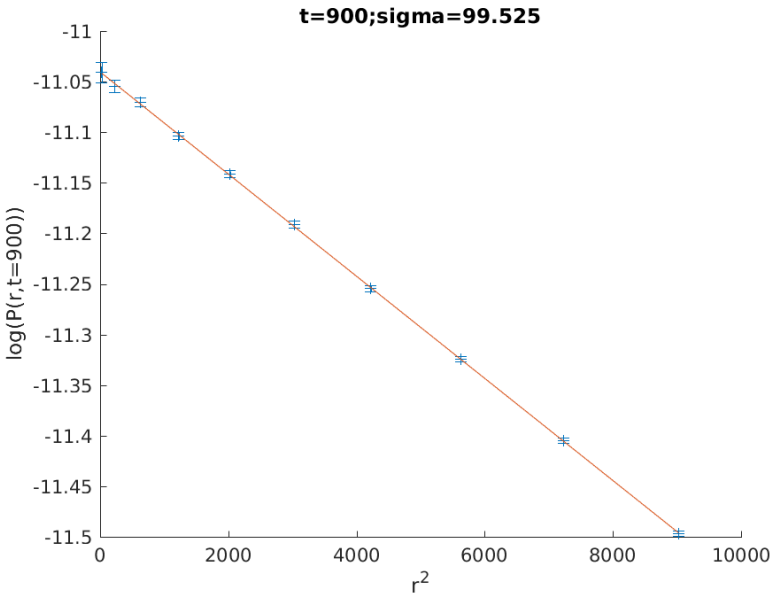


Figure 3: t=900



Firstly, we need to calculate our initial time point t0, since initial *sigma* is not zero here but the theoretical solution is. Because $\sigma(t=0) = \sigma_0 = \sqrt{2Dt0}$, $t0 = \frac{\sigma_0^2}{2D}$.

Finally, we get the picture 4, which compare our simulated σ with theoretical solution. We should notice that the theoretical solution must be translated left for $t0 = \frac{\sigma_0^2}{2D}$. So the simulation consists with theory quite well.

Figure 4: simulated σ with theoretical solution