Reproduction Report

> Reference: Simulation of a Brownian Particle in an Optical Trap [1]

Abstract

This report aims to replicate the simulation of a Brownian particle in an optical trap as described in the original study. The simulation setup and parameters were closely followed to ensure accuracy in reproduction. Results were compared with the original findings to validate the replication process. Challenges encountered during reproduction and potential areas for improvement are also discussed.

1. Introduction

The simulation of Brownian motion in an optical trap is of significant interest in various fields, including physics, chemistry, and biology. Understanding the behavior of particles in such traps provides insights into their dynamics and interactions with surrounding environments.

This report successfully replicated a substantial portion of the code and images from the referenced literature. Initially, the solution to the free diffusion equation with a white noise term was obtained, representing the stochastic process known as random walk. Subsequently, considering the Brownian motion of real particles and utilizing numerical treatments for the white noise term, the report discussed how particles transition from ballistic motion to diffusive behavior during free diffusion.

The study then delved into the motion of particles under the influence of external forces in an optical trap. Finally, it extended the analysis to the motion of particles subjected to more complex external forces, including practical applications such as photon force microscopy for measuring nanoscale forces exerted by biological molecules.

The subsequent sections of the report will replicate the results step by step structured by these contents and provide corresponding code for reference. Finally, the report will discuss and summarize any issues encountered during the replication process.

2. Random walks

The core task in this section is to treate the white noise term of the free diffusion equation within a finite difference framework. This provides numerical tools for subsequent handling of the stochastic differential equation of free diffusion. We can describe the motion of Brownian particles using the Langevin equation below.

$$m\ddot{x}(t) = -\gamma\dot{x}(t) + kx(t) + \sqrt{2k_BT\gamma}W(t) \tag{1}$$

where x is the particle position, m is the mass, γ is the friction coefficients, k is the trap stiffness, $\sqrt{2k_BT\gamma}W(t)$ is the Fluctuating force.

The finite difference method typically approximates $x(t), \dot{x}(t), \ddot{x}(t)$ using the following expression. In the report, the backward difference, which corresponds to the implicit Euler method, is employed.

$$x_i$$

$$(x_i - x_{i-1})/\Delta t$$

$$\frac{(x_i - x_{i-1})/\Delta t - (x_{i-1} - x_{i-2})/\Delta t}{\Delta t} = \frac{x_i - 2x_{i-1} + x_{i-2}}{\Delta t^2} \tag{2}$$

Discussing numerical treatment methods for white noise using the simple free diffusion equation below:

$$\dot{x}(t) = W(t) \tag{3}$$

where W_i approximates $W(t)$ with zero mean and variance $1/\Delta t$. $W_i = w_i/\sqrt{\Delta t}$ where w_i is the Gaussian random number sequence.

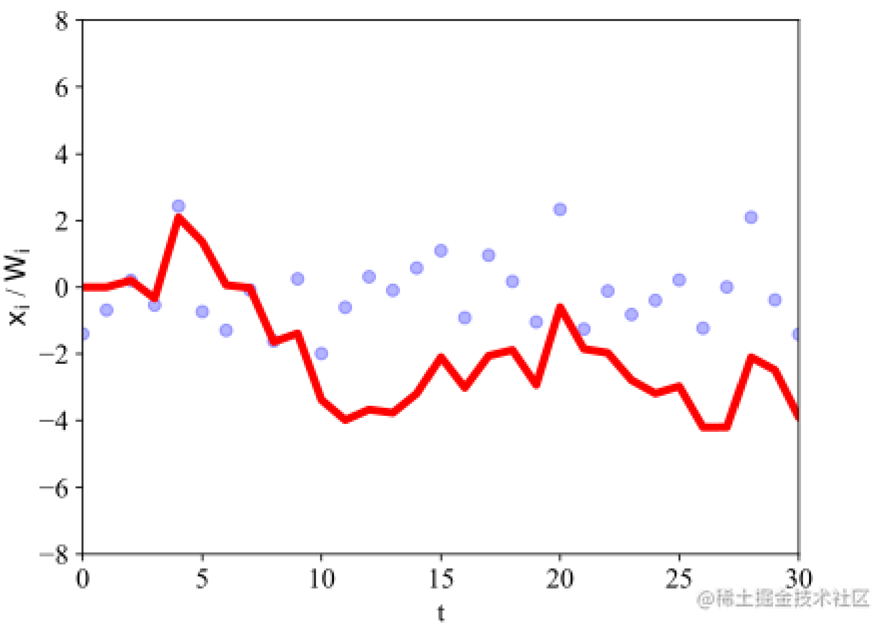
The finite difference form of this equation is:

$$\frac{x_i - x_{i-1}}{\Delta t} = \frac{w_i}{\sqrt{\Delta t}} \tag{4}$$

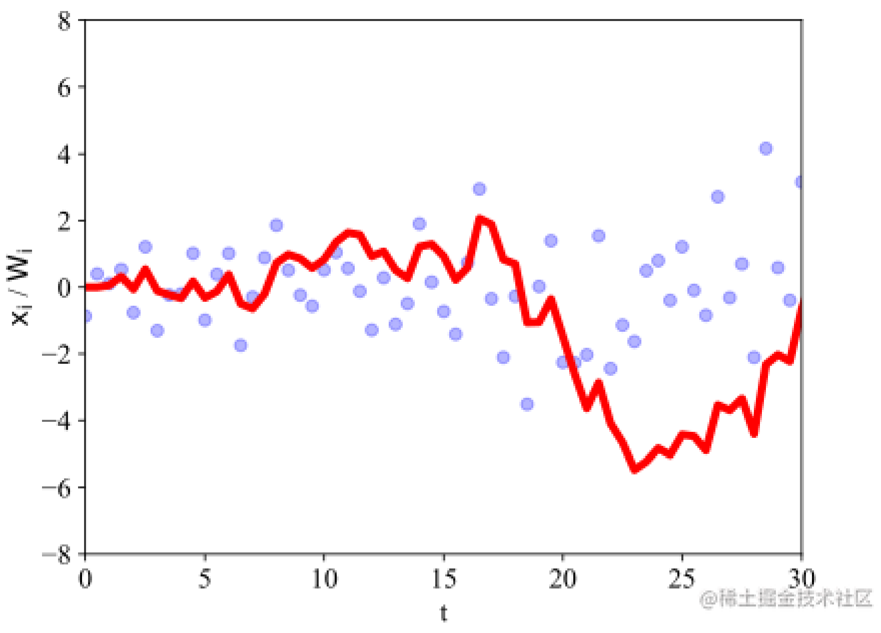
namely,

$$x_i = x_{i-1} + \sqrt{\Delta t} w_i \tag{5}$$

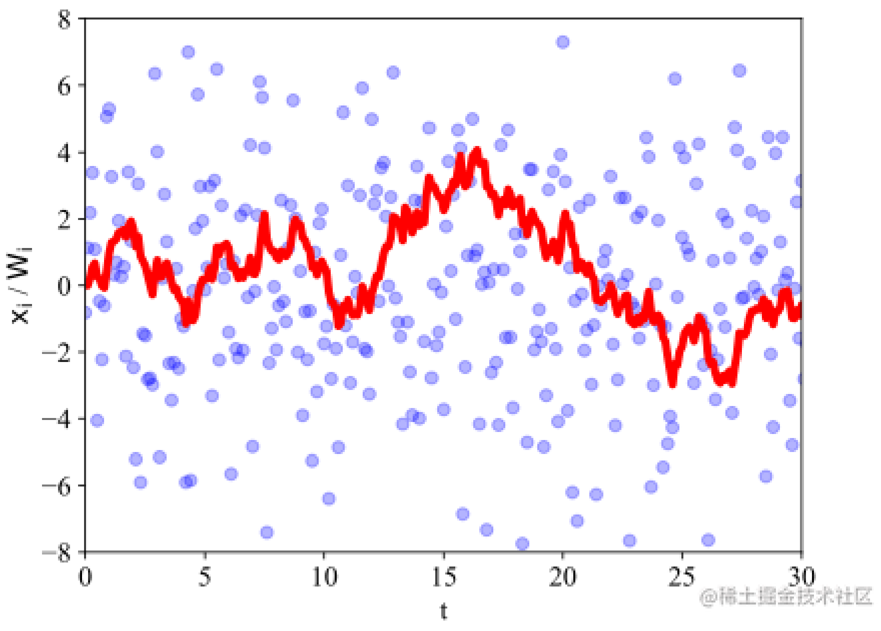
By solving the above finite difference equation, **Gaussian white noise and free diffusion trajectories** are demonstrated with $\Delta t = 1.0, 0.5, 0.1$.



$\Delta t = 1$



$\Delta t = 0.5$



$\Delta t = 0.1$

```
import numpy as np
import matplotlib.pyplot as plt


def random_walk(N, Dt, x1, x2, n):
    x = np.zeros(N)
    x[0] = x1
    x[1] = x2

    u1 = np.random.rand(n)
    u2 = np.random.rand(n)
    w = np.sqrt(-2 * np.log(u1)) * np.cos(2 * np.pi * u2)
    W = w / np.sqrt(Dt)
    for i in range(2, N):
        x[i] = x[i - 1] + np.sqrt(Dt) * w[i]

    t = np.arange(0, N * Dt, Dt)

    plt.figure()
    plt.xlim(0, 30)
    plt.ylim(-8, 8)
    plt.plot(t, x, linestyle="-", linewidth=4, color="red")
    plt.scatter(t, W, c="blue", alpha=0.3)
    plt.xlabel(
        r"t",
        fontdict={
            "family": "Times New Roman",
            "color": "black",
            "weight": "normal",
            "size": 14,
        },
    )
    plt.ylabel(
        r"$\mathrm{x_i}$ / $\mathrm{W_i}$",
        fontdict={
            "family": "Times New Roman",
            "color": "black",
            "weight": "normal",
            "size": 14,
        },
    )
    plt.xticks(fontname="Times New Roman", fontsize=14)
    plt.yticks(fontname="Times New Roman", fontsize=14)
    plt.show()

    return x, t


N = int(1e5)
Dt = 0.1
x1 = 0
x2 = 0
n = int(1e5)

random_walk(N, Dt, x1, x2, n)
```

3. Free diffusion

The main idea is to analysis the transition from ballistic to diffusive behavior induced by inertial effects at short time scales. Diffusion and friction coefficients satisfy the Einstein relation $\gamma D = k_B T$. As it is free diffusion, this section considers the Langevin equation without the restoring force term, as follows:

$$m\ddot{x}(t) = -\gamma\dot{x}(t) + \sqrt{2k_BT\gamma}W(t) \tag{6}$$

The particle parameters used in the simulation are as follows:

$$R = 1\mu m, m = 11pg, \eta = 0.001Nsm^{-2}, \gamma = 6\pi\eta R, T = 300K, \tau = m/\gamma = 0.6\mu s$$

3.1 Inertial regime

The corresponding finite difference format and solution are:

$$m\frac{x_i - 2x_{i-1} + xi - 2}{(\Delta t)^2} = -\gamma\frac{x_i - x_{i-1}}{\Delta t} + \sqrt{2k_BT\gamma}\frac{1}{\sqrt{\Delta t}}w_i \tag{7}$$

$$\begin{aligned} x_i = {} & \frac{2 + \Delta t(\gamma/m)}{1 + \Delta t(\gamma/m)}x_{i-1} - \frac{1}{1 + \Delta t(\gamma/m)}x_{i-2} \\ & + \frac{\sqrt{2k_BT\gamma}}{m[1 + \Delta t(\gamma/m)]}(\Delta t)^{3/2}w_i \end{aligned} \tag{8}$$

where $\tau = m/\gamma$ is the momentum relaxation time. This time scale represents for the transition from smooth ballistic behavior to diffusive behavior.

3.2 Diffusive regime

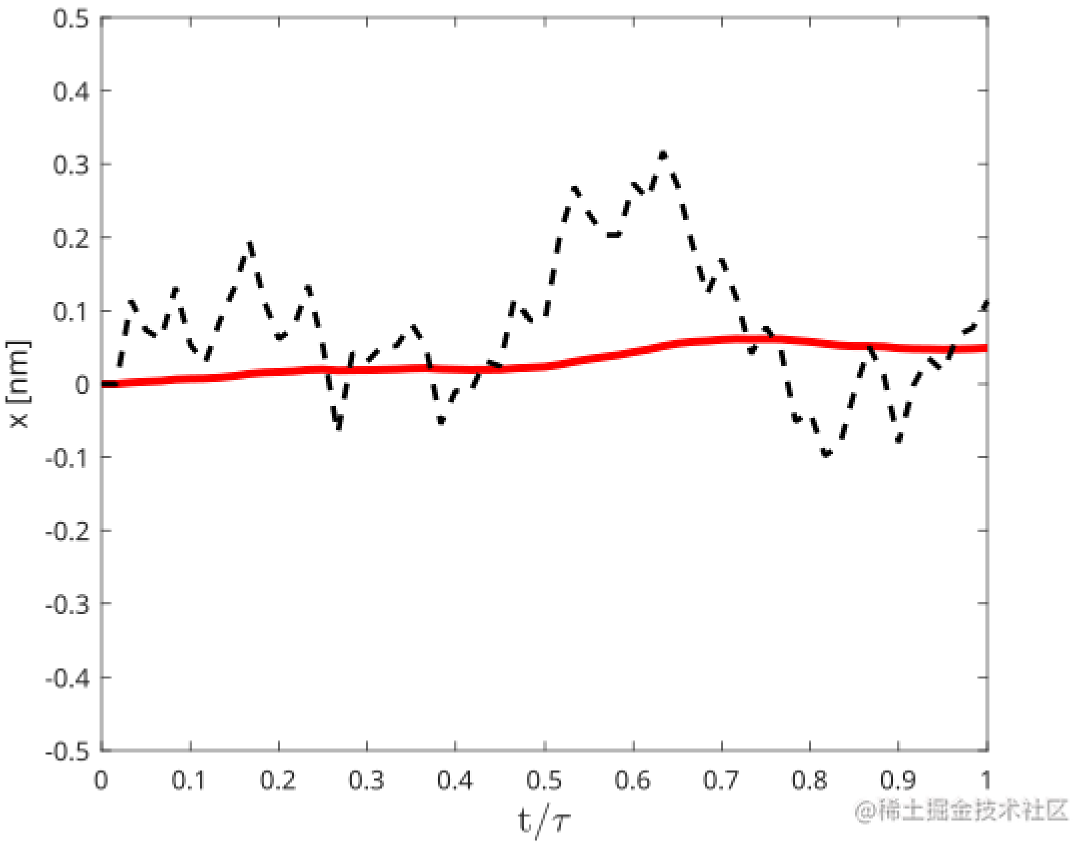
Due to the small time scale, neglecting the inertial term yields:

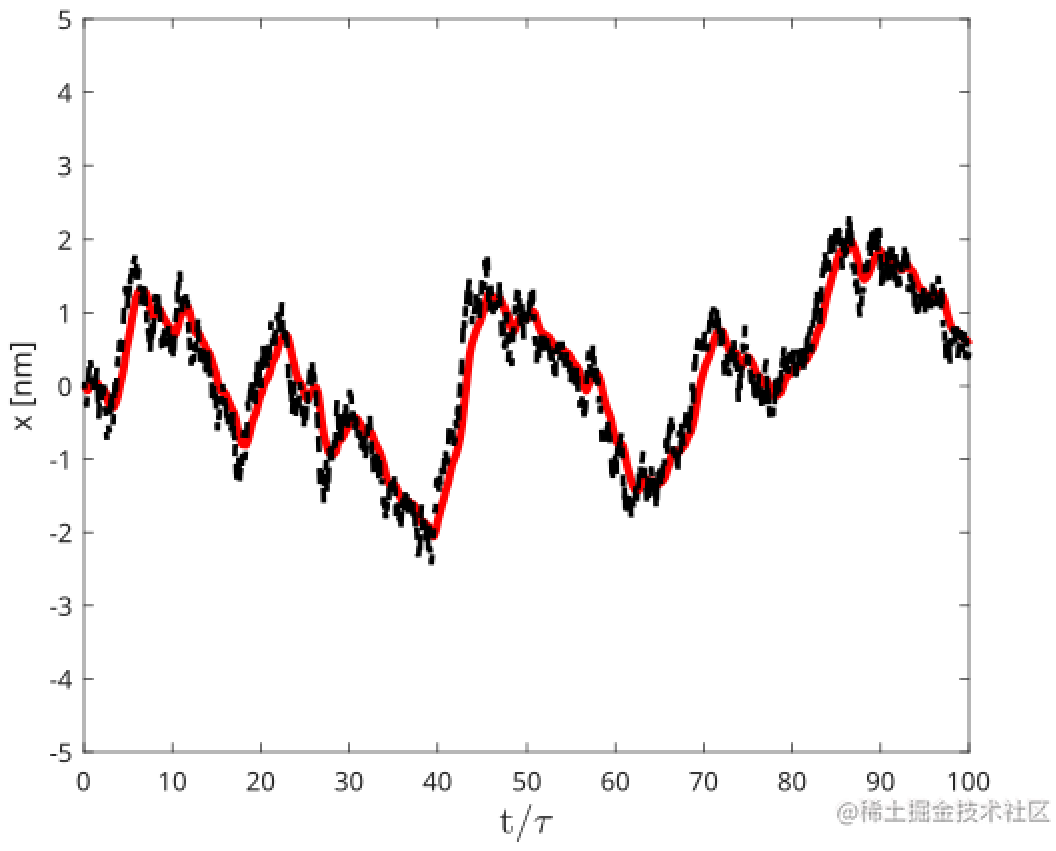
$$\dot{x}(t) = \sqrt{2D}W(t) \tag{9}$$

The corresponding finite difference form is:

$$x_i = x_{i-1} + \sqrt{2D\Delta t}w_i \tag{10}$$

Trajectories in inertial and diffusive regimes can be plotted as:





```
function [x, t] = inertial_diffusive(N, Dt, x1, x2, R, T, eta)
  kB = 1.38e-23;  % Boltzmann constant [J/K]
  gamma = 6 * pi * R * eta; % friction coefficient
  m = 11e-15; %[Kg]
  tau = 0.6e-6; %[s]
  x(1) = x1; x(2) = x2; % initial conditions
  y(1) = x1; y(2) = x2;
  D = kB*T/gamma; % diffusion coefficient
  nm = 1e-9;


  for i = 3:1:N
      w_i = randn();
    x(i) = (2 + Dt * gamma/m)/(1 + Dt * gamma/m) * x(i - 1) ...
        - 1/(1 + Dt * gamma/m) * x(i - 2) ...
        + sqrt(2 * kB * T * gamma)/(m + Dt * gamma) * w_i * power(Dt, 1.5);
    y(i) = y(i-1) + sqrt(2*D*Dt)*w_i;
  end
  t = [0:Dt:(N - 1) * Dt];


  plot(t/tau, x/nm,'-r', 'Linewidth', 3)
  hold on;
  plot(t/tau,y/nm,'--black', 'Linewidth', 2)
  xlabel('t/$\tau$', 'Interpreter', 'latex', 'FontSize', 14);
  ylabel('x [nm]')
  xlim([0, 1]);
  ylim([-0.5, 0.5]);
%   xlim([0, 100]);
%   ylim([-5, 5]);
```

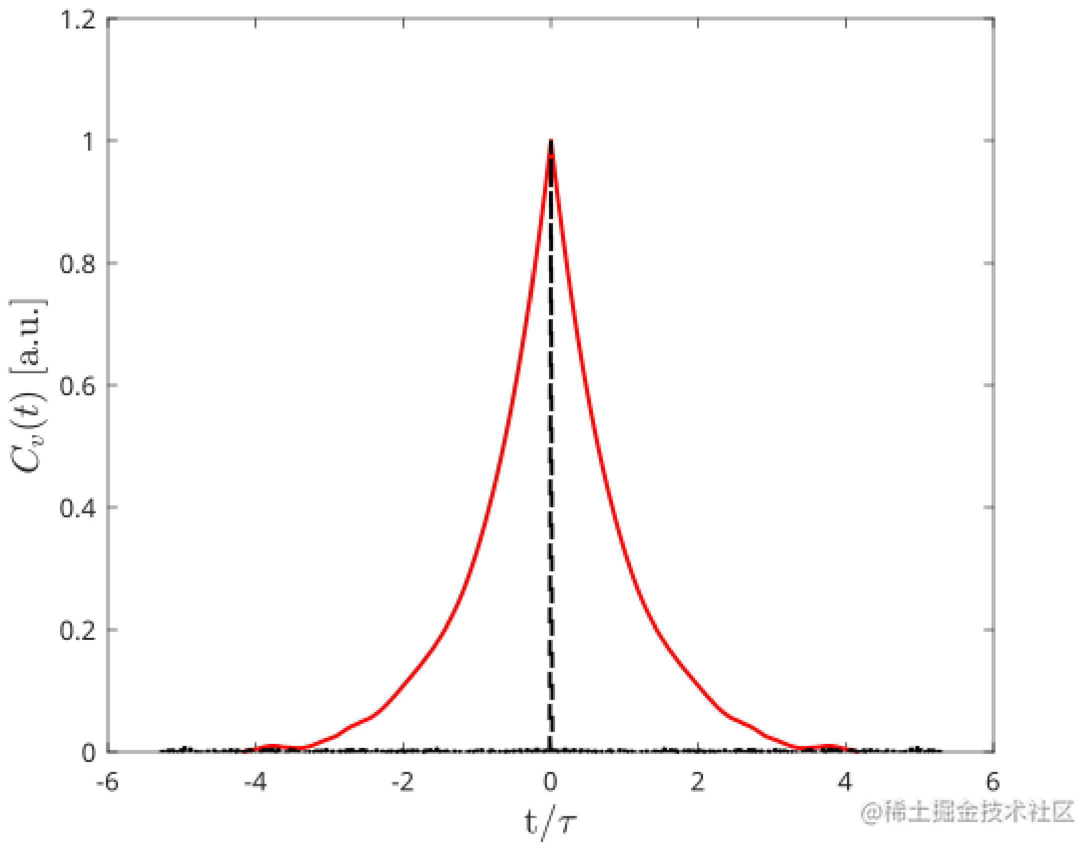
```
  hold on;
```

3.3 Velocity autocorrelation function

The velocity autocorrelation function is a time measure that is independent of the particle's initial velocity. The definition and corresponding finite difference discrete format are as follows:

$$C_v(t) = \overline{v(t'+t)v(t')} \tag{11}$$

$$C_{v,n} = \overline{v_{i+n}v_i} \tag{12}$$



```
function [rx,ry,s]=vacf(x,y,Dt)
vx = (x(2:end)-x(1:end-1))/Dt; % average speed
vy = (y(2:end)-y(1:end-1))/Dt; % average speed
rx = xcorr(vx,ceil(sqrt(length(x))),'unbiased');
ry = xcorr(vy,ceil(sqrt(length(y))),'unbiased');
% normalize the velocity autocorrelation function
rx = rx / rx(ceil(end/2));
ry = ry / ry(ceil(end/2));
% define the delay time length
num_lags = length(rx) - 1;
s = Dt*(-num_lags/2:1:num_lags/2);
tau = 0.6e-6;

figure
plot(s/tau,rx,'-r', 'Linewidth', 1.5)
hold on;
plot(s/tau,ry,'--black', 'Linewidth', 1.5)
xlabel('t/$\mathrm{\tau}$', 'Interpreter', 'latex', 'FontSize', 14)
ylabel('$C_v(t)$ [a.u.]', 'Interpreter', 'latex', 'FontSize', 14)
xlim([-6, 6]);
```

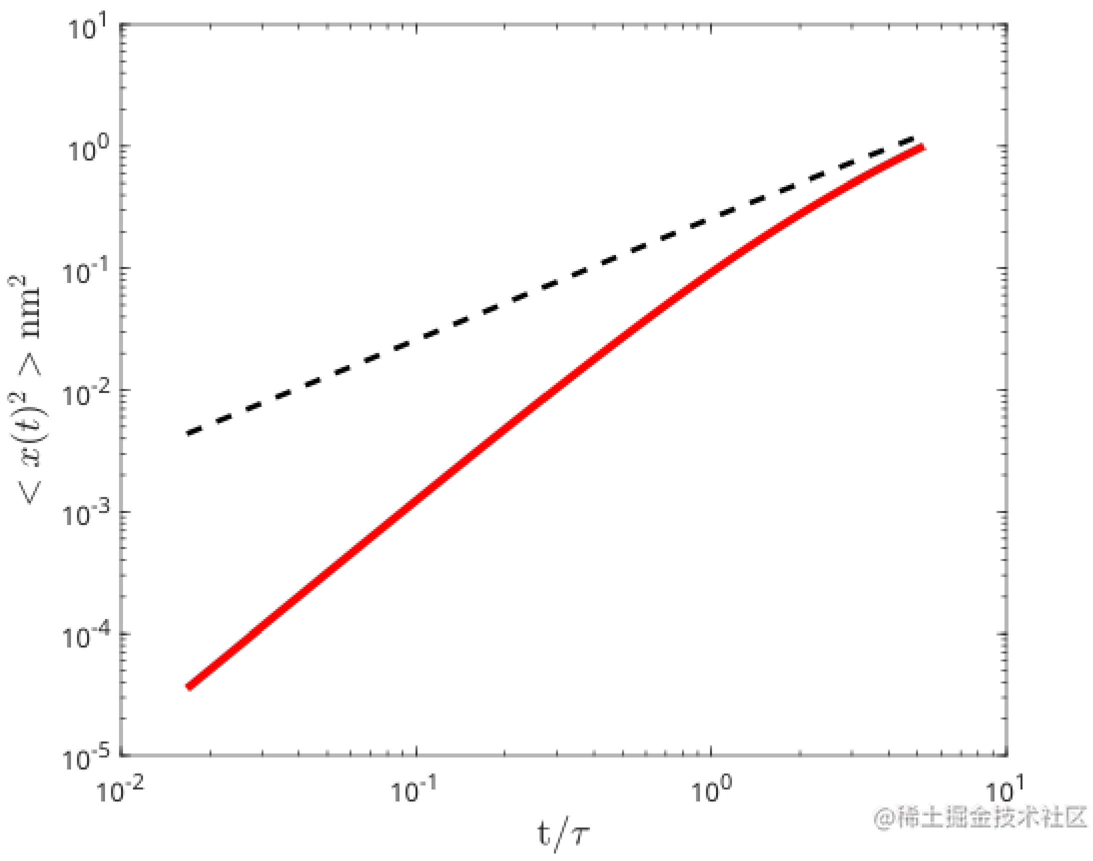
```
ylim([0, 1.2]);
```

3.4 Mean square displacement

Mean square displacement quantifies how a particle moves from its initial position. The definition and corresponding finite difference discrete format are as follows:

$$\langle x(t)^2 \rangle = \overline{[x(t'+t) - x(t')]^2} \tag{13}$$

$$\langle x_n^2 \rangle = \overline{[x_{i+n} - x_i]^2} \tag{14}$$



```
function [msdx,msdy,sx,sy] = MSD(x,y,Dt)
  for n = 0:1:sqrt(length(x))
      msdx(n + 1) = mean((x(n + 1:end) - x(1:end - n)).^2);
  end
  sx = Dt * [0:1:length(msdx) - 1];
  for n = 0:1:sqrt(length(y))
      msdy(n + 1) = mean((y(n + 1:end) - y(1:end - n)).^2);
  end
  sy = Dt * [0:1:length(msdy) - 1];
  nm2 = 1e-18;
  tau = 0.6e-6;

  figure
  loglog(sx/tau, msdx/nm2, '-r', 'Linewidth', 3)
  hold on;
  loglog(sy/tau, msdy/nm2, '--black', 'Linewidth', 2)
  xlabel('t/$\tau$', 'Interpreter', 'latex', 'FontSize', 14);
  ylabel('$<x(t)^2> \mathrm{nm^2}$', 'Interpreter', 'latex', 'FontSize', 14)
```

4. Trapped particle

In this section, we are going to talk about the Influence of optical traps on particle motion.

The Langevin equation in three-dimensional space is as follows:

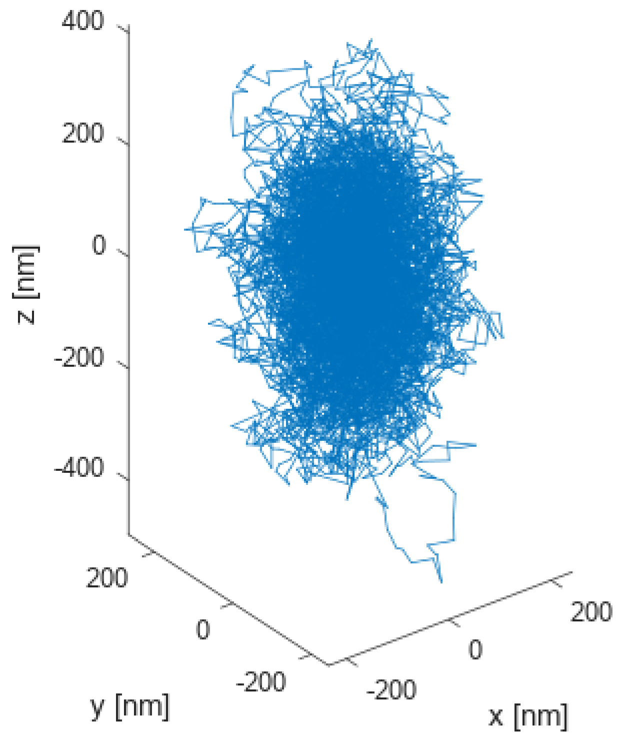
$$\vec{r_i} = \vec{r_{i-1}} - \frac{1}{\gamma}\vec{k} \cdot \vec{r_{i-1}}\Delta t + \sqrt{2D\Delta t}\vec{w_i} \tag{16}$$

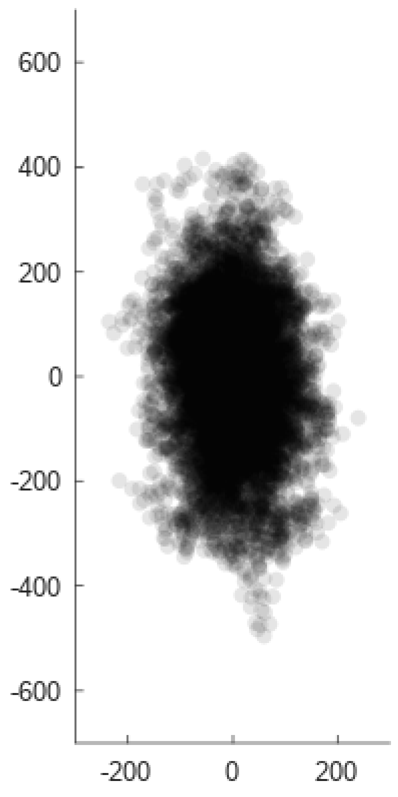
where $k_x = k_y = 1.0 \times 10^{-6} N/m$ and $k_z = 0.2 \times 10^{-6} N/m$

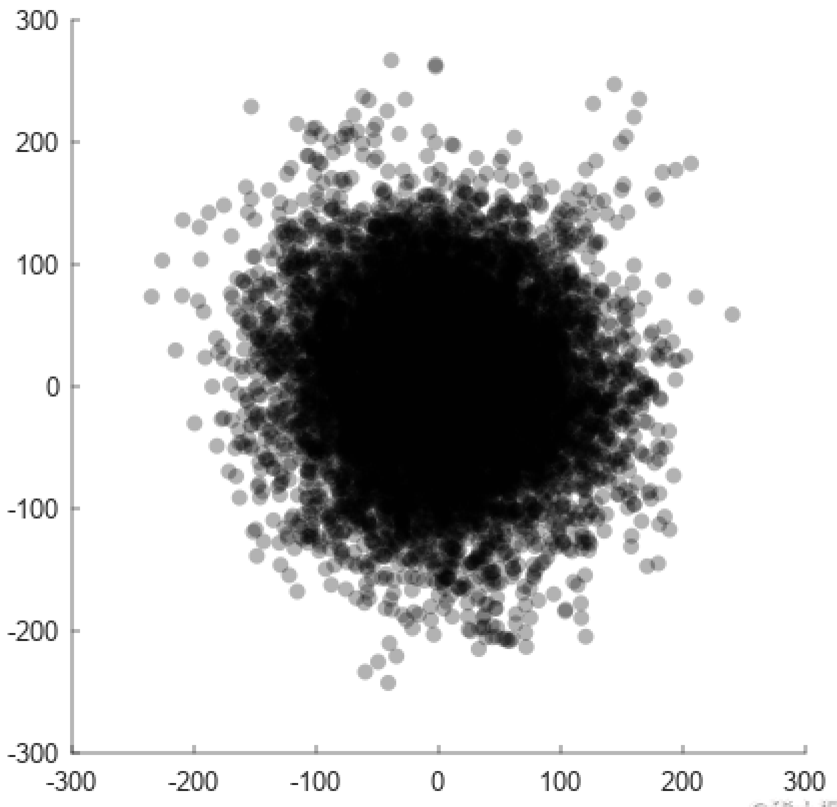
Position autocorrelation function describes how particles enter the trap.

$$C_x(t) = \overline{x(t' + t)x(t')} \tag{17}$$

Trajectories of Brownian particles in optical trap and corresponding **projection** on xz plane and xy plane:

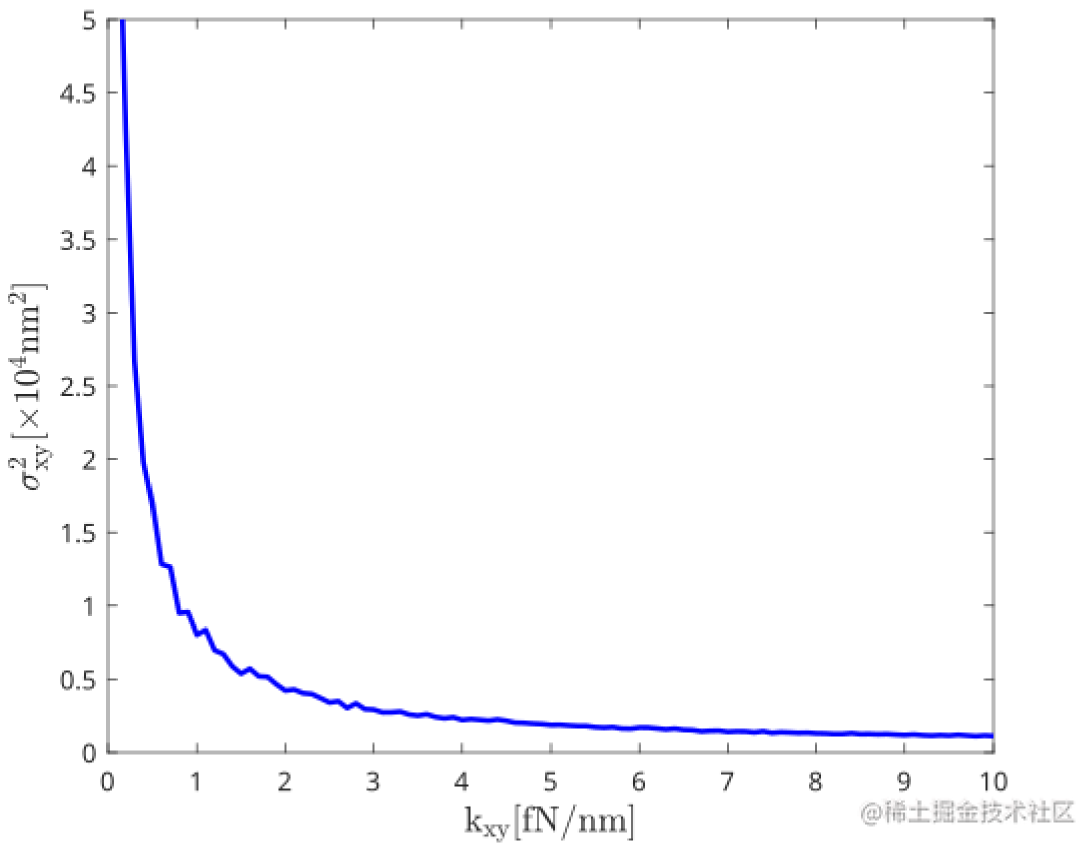






```
function [x,y,z,t]=...
  trapped(N,Dt,x1,y1,z1,R,T,eta,kx,ky,kz)

  kB = 1.38e-23;  % Boltzmann constant [J/K]
  gamma = 6*pi*R*eta; % friction coefficient
  D = kB*T/gamma; % diffusion coefficient
  x(1)=x1;y(1)=y1;z(1)=z1;  % initial condition
  for i = 2:1:N
    % Deterministic step
    x(i) = x(i-1) - kx*Dt/gamma*x(i-1);
    y(i) = y(i-1) - ky*Dt/gamma*y(i-1);
    z(i) = z(i-1) - kz*Dt/gamma*z(i-1);
    % Diffusive step
    x(i) = x(i) + sqrt(2*D*Dt)*randn();
    y(i) = y(i) + sqrt(2*D*Dt)*randn();
    z(i) = z(i) + sqrt(2*D*Dt)*randn();
  end
  t = [0:Dt:(N-1)*Dt];
  nm = 1e-9;


  figure
  plot3(x/nm,y/nm,z/nm);
  xlabel('x [nm]')
  ylabel('y [nm]')
  zlabel('z [nm]')
  axis equal

  % plot projection on x-y plane
  figure
  scatter(x/nm, y/nm, 'filled', 'MarkerFaceColor', 'k', 'MarkerEdgeColor',
'none', 'MarkerFaceAlpha', 0.3);
  axis equal
  xlim([-300, 300])
  ylim([-300, 300])


  % plot projection on x-z plane
  figure
  scatter(x/nm, z/nm, 'filled', 'MarkerFaceColor', 'k', 'MarkerEdgeColor',
'none', 'MarkerFaceAlpha', 0.1);
  axis equal
  xlim([-300, 300])
  ylim([-700, 700])
```

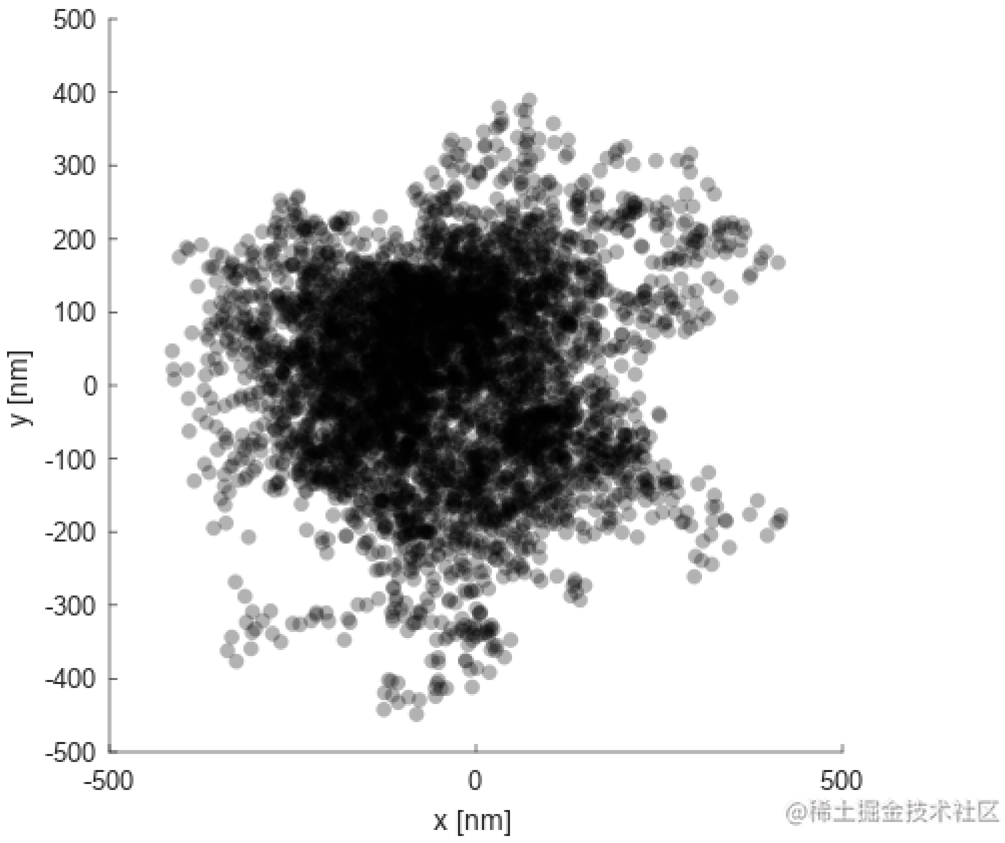
Displacement variance around the trap center of the particle on xy plane:



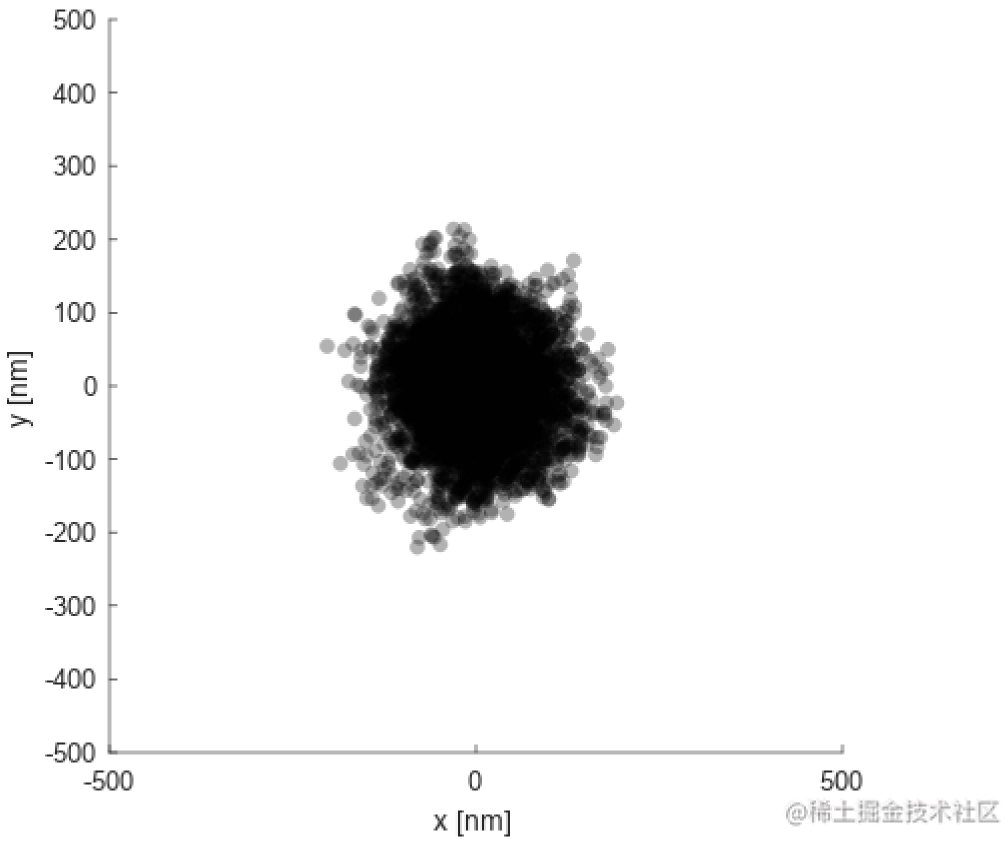
```
function [x,y,z,t]=...
  trapped(N,Dt,x1,y1,z1,R,T,eta,kx,ky,kz)

  kB = 1.38e-23;  % Boltzmann constant [J/K]
  gamma = 6*pi*R*eta; % friction coefficient
  D = kB*T/gamma; % diffusion coefficient

  x(1)=x1;y(1)=y1;z(1)=z1;  % initial condition
  for i = 2:1:N
    % Deterministic step
    x(i) = x(i-1) - kx*Dt/gamma*x(i-1);
    y(i) = y(i-1) - ky*Dt/gamma*y(i-1);
    z(i) = z(i-1) - kz*Dt/gamma*z(i-1);
    % Diffusive step
    x(i) = x(i) + sqrt(2*D*Dt)*randn();
    y(i) = y(i) + sqrt(2*D*Dt)*randn();
    z(i) = z(i) + sqrt(2*D*Dt)*randn();
  end
  t = [0:Dt:(N-1)*Dt];
```

```
function [ ] = sigmaxy2_kxy()
sigmaxy2 = zeros();
for i = 1:1:100
    kx = i * 1e-7;
    ky = kx;
    kz = 0.2e-6;
    N = 5e3;
    Dt = 1e-3;
    x1 = 0;
    y1 = 0;
    z1 = 0;
    R = 1e-6;
    T = 300;
    eta = 0.001;
    nm = 1e-9;
    [x,y,z,t] = trapped(N,Dt,x1,y1,z1,R,T,eta,kx,ky,kz);
    distance = x.^2 + y.^2;
    sigmaxy2(i) = sum(distance)/N/(nm^2);
end
s = [0.1:0.1:10];
plot(s, sigmaxy2/(1e4),'-b', 'Linewidth', 2)
ylim([0, 5])
ylabel('$\mathrm{\sigma_{xy}^2[\times 10^4 nm^2]}$', 'Interpreter', 'latex',
'FontSize', 14);
xlabel('$\mathrm{k_{xy}[fN/nm]}$', 'Interpreter', 'latex', 'FontSize', 14')
end
```

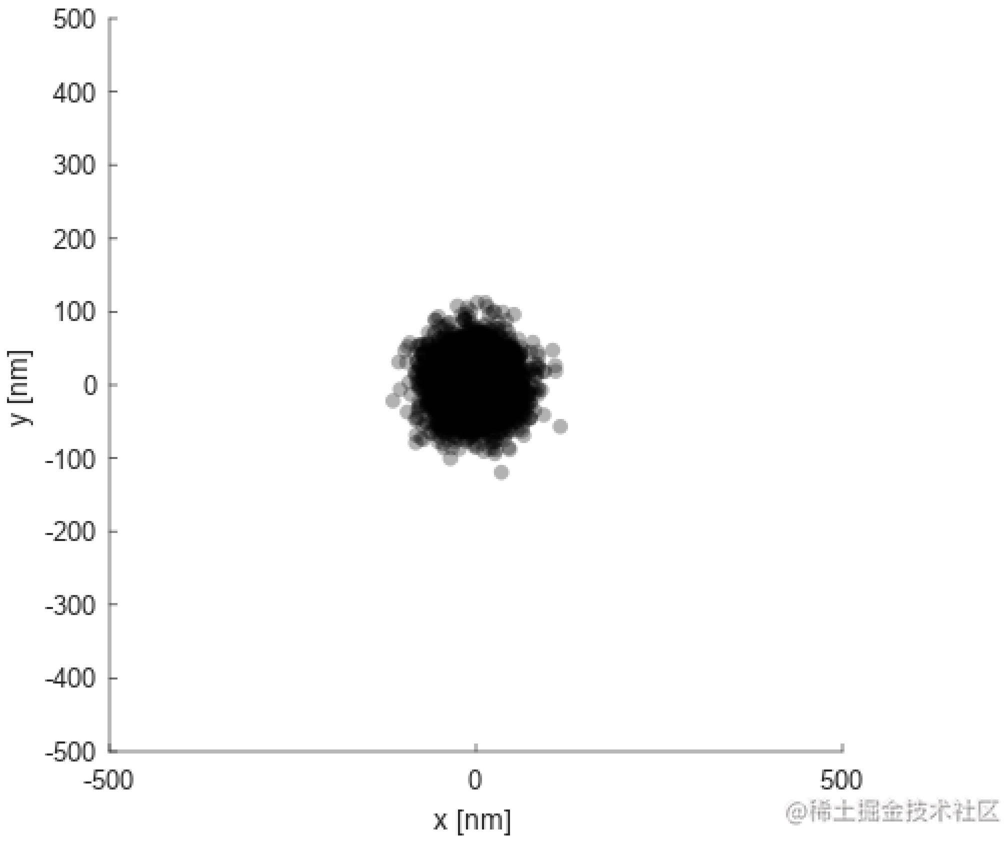
Particle distribution on xy plane with
$k_{xy} = 0.2fN/nm, k_{xy} = 1.0fN/nm, k_{xy} = 5.0fN/nm$:



$k_{xy} = 0.2fN/nm$



$$k_{xy} = 1.0fN/nm$$

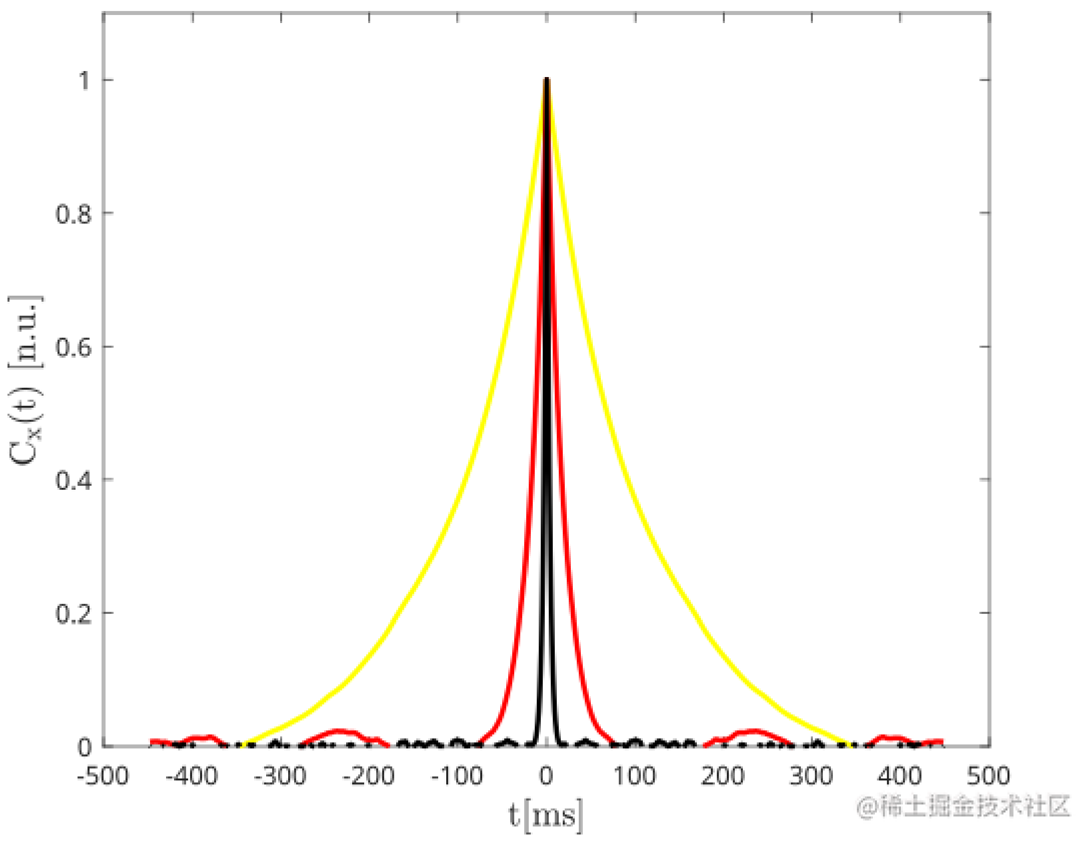


$$k_{xy} = 5.0fN/nm$$

```
function [x,y,z,t]=...
  trapped(N,Dt,x1,y1,z1,R,T,eta,kx,ky,kz)
```

```
kB = 1.38e-23;  % Boltzmann constant [J/K]
gamma = 6*pi*R*eta; % friction coefficient
D = kB*T/gamma; % diffusion coefficient

x(1)=x1;y(1)=y1;z(1)=z1;  % initial condition
for i = 2:1:N
  % Deterministic step
  x(i) = x(i-1) - kx*Dt/gamma*x(i-1);
  y(i) = y(i-1) - ky*Dt/gamma*y(i-1);
  z(i) = z(i-1) - kz*Dt/gamma*z(i-1);
  % Diffusive step
  x(i) = x(i) + sqrt(2*D*Dt)*randn();
  y(i) = y(i) + sqrt(2*D*Dt)*randn();
  z(i) = z(i) + sqrt(2*D*Dt)*randn();
end
t = [0:Dt:(N-1)*Dt];
nm = 1e-9;


% plot projection on x-y plane
figure
scatter(x/nm, y/nm, 'filled', 'MarkerFaceColor', 'k', 'MarkerEdgeColor',
'none', 'MarkerFaceAlpha', 0.3);
axis equal
xlim([-500, 500])
ylim([-500, 500])
xlabel('x [nm]')
ylabel('y [nm]')
```

Position autocorrelation function of the trapped particle is shown as follows:

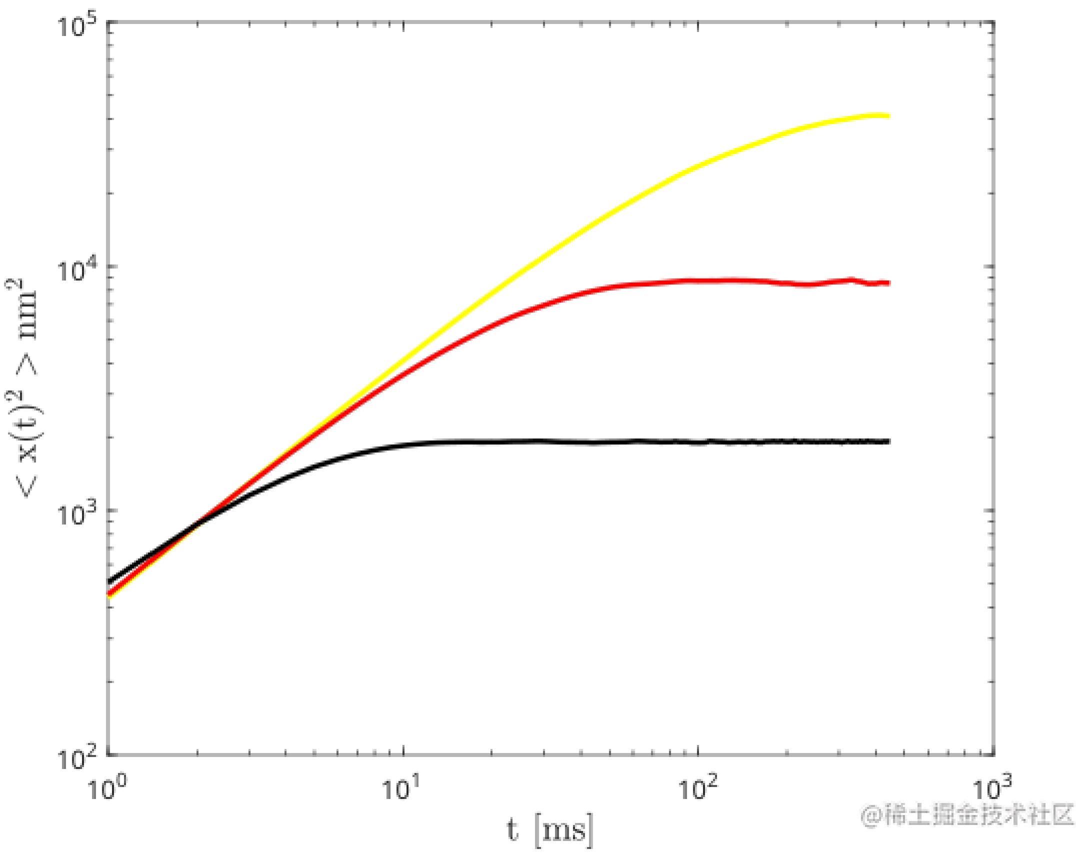


```
function [x,t]=...
  forced_trap(N,Dt,x1,R,T,eta,kx)

  kB = 1.38e-23;  % Boltzmann constant [J/K]
  gamma = 6*pi*R*eta; % friction coefficient
  D = kB*T/gamma; % diffusion coefficient

  %T = 300;         % absolute temprature [K]
  %gamma = 0.001;  % friction coefficient [kg/s]
  m = 1e-15;      % particle mass [kg]
  nm = 1e-9;
  ms = 1e-3;

  x(1)=x1;  % initial condition
  t = [0:Dt:(N-1)*Dt];


  for i = 2:1:N
    % External forces
    F_xDt = -kx*Dt*x(i-1);

    % Deterministic step
    x(i) = x(i-1) + F_xDt/gamma;

    % Diffusive step
    x(i) = x(i) + sqrt(2*D*Dt)*randn();
  end
```

```
function [ ] = pacf_tri( x,y,z,Dt )
rx = xcorr(x,ceil(sqrt(length(x))),'unbiased');
maxProbability = max(abs(rx));
% normalize the velocity autocorrelation function
rx = rx / rx(ceil(end/2));
% define the delay time length
num_lags = length(rx) - 1;
sx = Dt*(-num_lags/2:1:num_lags/2);

ry = xcorr(y,ceil(sqrt(length(y))),'unbiased');
maxProbability = max(abs(ry));
% normalize the velocity autocorrelation function
ry = ry / ry(ceil(end/2));
% define the delay time length
num_lags = length(ry) - 1;
sy = Dt*(-num_lags/2:1:num_lags/2);


rz = xcorr(z,ceil(sqrt(length(z))),'unbiased');
maxProbability = max(abs(rz));
% normalize the velocity autocorrelation function
rz = rz / rz(ceil(end/2));
% define the delay time length
num_lags = length(rz) - 1;
sz = Dt*(-num_lags/2:1:num_lags/2);
```

```
tau = 0.6e-6;
ms = 1e-3;
nm = 1e-9;

figure
plot(sx/ms,rx,'-y', 'Linewidth', 2)
hold on;
plot(sy/ms,ry,'-r', 'Linewidth', 2)
hold on;
plot(sz/ms,rz,'-black', 'Linewidth', 2)
hold on;
xlabel('$\mathrm{t [ms]}$', 'Interpreter', 'latex', 'FontSize', 14)
ylabel('$\mathrm{C_{x}(t)}$ [n.u.]', 'Interpreter', 'latex', 'FontSize', 14)
xlim([-500, 500]);
ylim([0, 1.1]);
hold on;

end
```

and the **mean square displacement** is demonstrated in the following:



```
function [x,t]=...
  forced_trap(N,Dt,x1,R,T,eta,kx)

  kB = 1.38e-23;  % Boltzmann constant [J/K]
  gamma = 6*pi*R*eta; % friction coefficient
  D = kB*T/gamma; % diffusion coefficient

  %T = 300;         % absolute temprature [K]
  %gamma = 0.001;   % friction coefficient [kg/s]
  m = 1e-15;       % particle mass [kg]
  nm = 1e-9;
  ms = 1e-3;

  x(1)=x1;  % initial condition
  t = [0:Dt:(N-1)*Dt];


  for i = 2:1:N
    % External forces
    F_xDt = -kx*Dt*x(i-1);

    % Deterministic step
    x(i) = x(i-1) + F_xDt/gamma;

    % Diffusive step
    x(i) = x(i) + sqrt(2*D*Dt)*randn();
  end
```

```
function [] = MSD_tri(x,y,z,Dt)
  for n = 0:1:sqrt(length(x))
      msdx(n + 1) = mean((x(n + 1:end) - x(1:end - n)).^2);
  end
  sx = Dt * [0:1:length(msdx) - 1];
  for n = 0:1:sqrt(length(y))
      msdy(n + 1) = mean((y(n + 1:end) - y(1:end - n)).^2);
  end
  sy = Dt * [0:1:length(msdy) - 1];
  for n = 0:1:sqrt(length(z))
      msdz(n + 1) = mean((z(n + 1:end) - z(1:end - n)).^2);
  end
  sz = Dt * [0:1:length(msdz) - 1];
  nm2 = 1e-18;
  tau = 0.6e-6;
  ms = 1e-3;

  figure
  loglog(sx/ms, msdx/nm2, '-y', 'Linewidth', 2)
  hold on;
  loglog(sy/ms, msdy/nm2, '-r', 'Linewidth', 2)
  hold on;
  loglog(sz/ms, msdz/nm2, '-black', 'Linewidth', 2)
  xlabel('t [ms]', 'Interpreter', 'latex', 'FontSize', 14);
  ylabel('$\mathrm{<x(t)^2> nm^2}$', 'Interpreter', 'latex', 'FontSize', 14)

end
```

5. Complex external forces

5.1 Simple optical trap

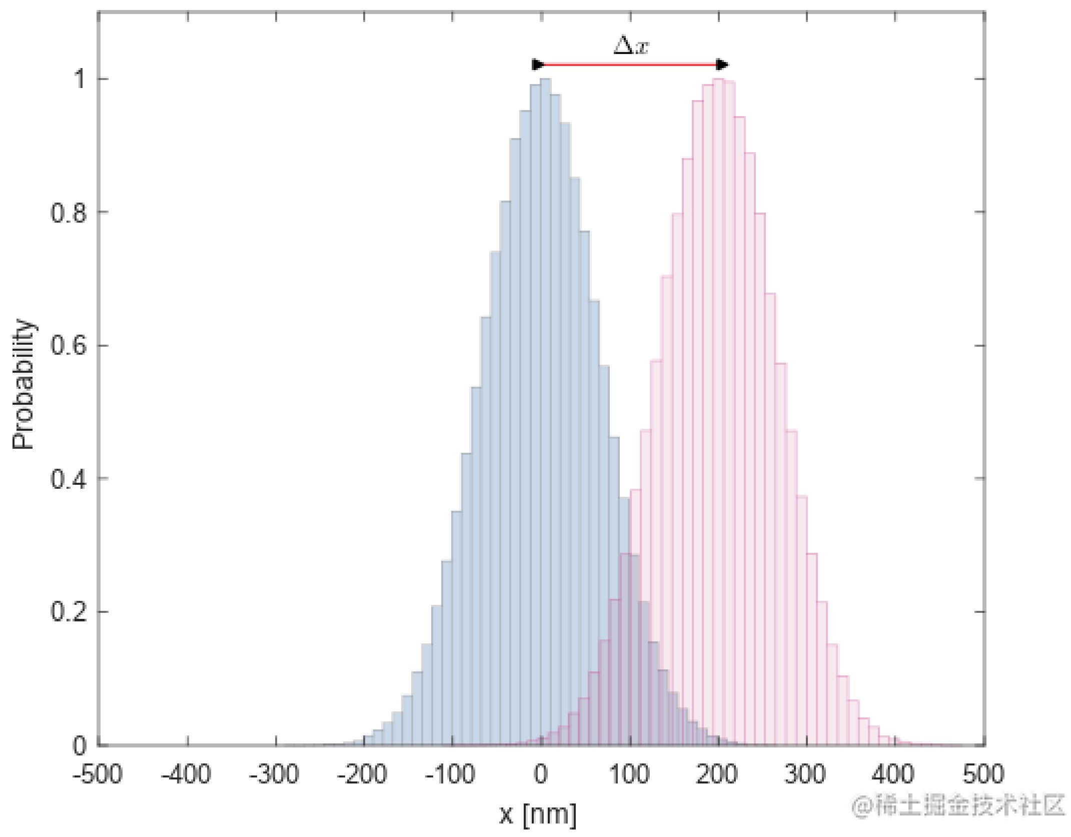
Equation (1) can be generalized as:

$$\dot{x}(t) = \frac{1}{\gamma} F(x(t), t) + \sqrt{2D} W(t) \tag{18}$$

Let $F(x(t), t) = -kx(t)$, then we have equation (1).

5.2 Photonic force microscopy

Let $F(x(t), t) = -kx(t) + Fc(t) * h(t)$, $h(t)$ is the Heaviside function. Then $F_c = k\Delta x$ can be measured.



```
function [x,y,t]=...
  forced_trap(N,Dt,x1,y1,R,T,eta,a,b,kx)

  kB = 1.38e-23;  % Boltzmann constant [J/K]
  gamma = 6*pi*R*eta; % friction coefficient
  D = kB*T/gamma; % diffusion coefficient

  %T = 300;          % absolute temprature [K]
  %gamma = 0.001;   % friction coefficient [kg/s]
  m = 1e-15;        % particle mass [kg]
  nm = 1e-9;
  ms = 1e-3;

  c = sqrt(2 * kB * T * gamma);
  f = 1 / (2 * pi) * sqrt(kB * T * gamma / m);
  x(1)=x1;  % initial condition
  y(1)=y1;
  k = kx;
  omega = 132.6;
  t = [0:Dt:(N-1)*Dt];


  for i = 2:1:N
      % External forces
%     F_xDt = (-a * x(i - 1)^3 + b * x(i - 1) + ...
%         c * sin(2 * pi * f * t(i - 1))) * Dt;

%     F_xDt = (-a * x(i - 1)^3 + b * x(i - 1)) * Dt;

    F_xDt = -kx*Dt*x(i-1);

%     Fc = 200e-15;
%     F_xDt = -kx*Dt*x(i-1) + Fc*Dt;


    % Deterministic step
    x(i) = x(i-1) + F_xDt/gamma;

%     x(i) = x(i-1) - (k*x(i-1)+gamma*omega*y(i-1))*Dt/gamma;
%     y(i) = y(i-1) - (-gamma*omega*x(i-1) + k*y(i-1))*Dt/gamma;

    % Diffusive step
    x(i) = x(i) + sqrt(2*D*Dt)*randn();

%     x(i) = x(i) + sqrt(2*D*Dt)*randn();
%     y(i) = y(i) + sqrt(2*D*Dt)*randn();
  end

  figure
  plot(t,x/nm)
  xlabel('t [s]')
  ylabel('x [nm]')
%   plot(x/nm, y/nm)
%   xlabel('x [nm]')
%   ylabel('y [nm]')
  xlim([0,200])
```

```
function [] = dual_hist(x,x_Dt)

% Plot the histogram.
% Optional: Adjust the number of bins
% Plot the histogram and adjust the edge transparency
nm = 1e-9;
h1 = histogram(x/nm, 'Normalization', 'countdensity', ...
    'EdgeColor', [0.5 0.5 0.5], 'EdgeAlpha', 0.5, ...
    'FaceColor', [0.3 0.5 0.7], 'FaceAlpha', 0.3, ...
    'NumBins', 50);
% Find the maximum probability
maxProbability_h1 = max(h1.Values);
```

```
% Normalize the entire histogram
h1.BinCounts = h1.BinCounts / maxProbability_h1;
hold on;

h2 = histogram(x_Dt/nm, 'Normalization', 'countdensity', ...
    'EdgeColor', [0.8 0.3 0.6], 'EdgeAlpha', 0.5, ...
    'FaceColor', [0.7 0.2 0.4], 'FaceAlpha', 0.1, ...
    'NumBins', 50);
% Find the maximum probability
maxProbability_h2 = max(h2.Values);

% Normalize the entire histogram
h2.BinCounts = h2.BinCounts / maxProbability_h2;
hold off;

% Add title and labels
xlabel('x [nm]');
ylabel('Probability');
xlim([-500, 500]);
ylim([0, 1.1])
hold on;
end
```

5.3 Rotational force field

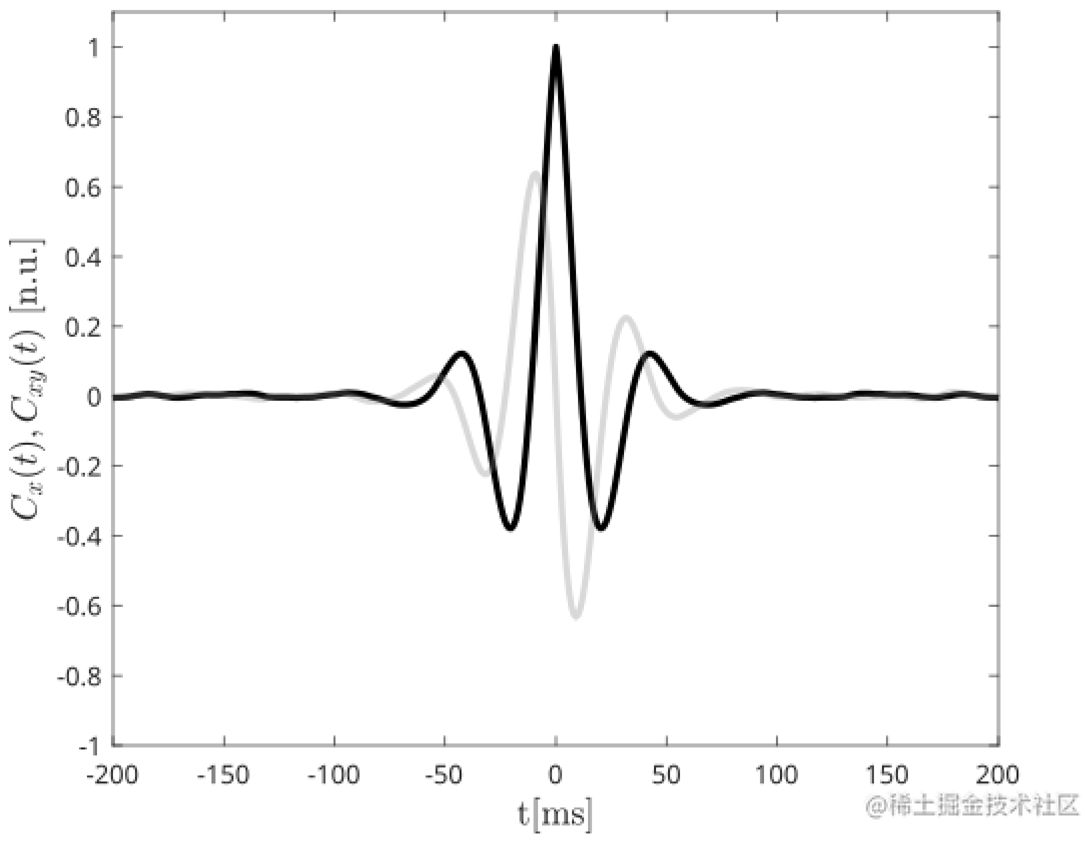
Let

$$\vec{F}(x,y) = -\begin{bmatrix} k & \gamma\Omega \\ -\gamma\Omega & k \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} \tag{19}$$

where Ω represents the rotation component.

To describe the motion correlation on x and y direction, **position cross-correlation** function is expressed as:

$$C_{xy}(t) = \overline{x(t'+t)y(t')} \tag{20}$$



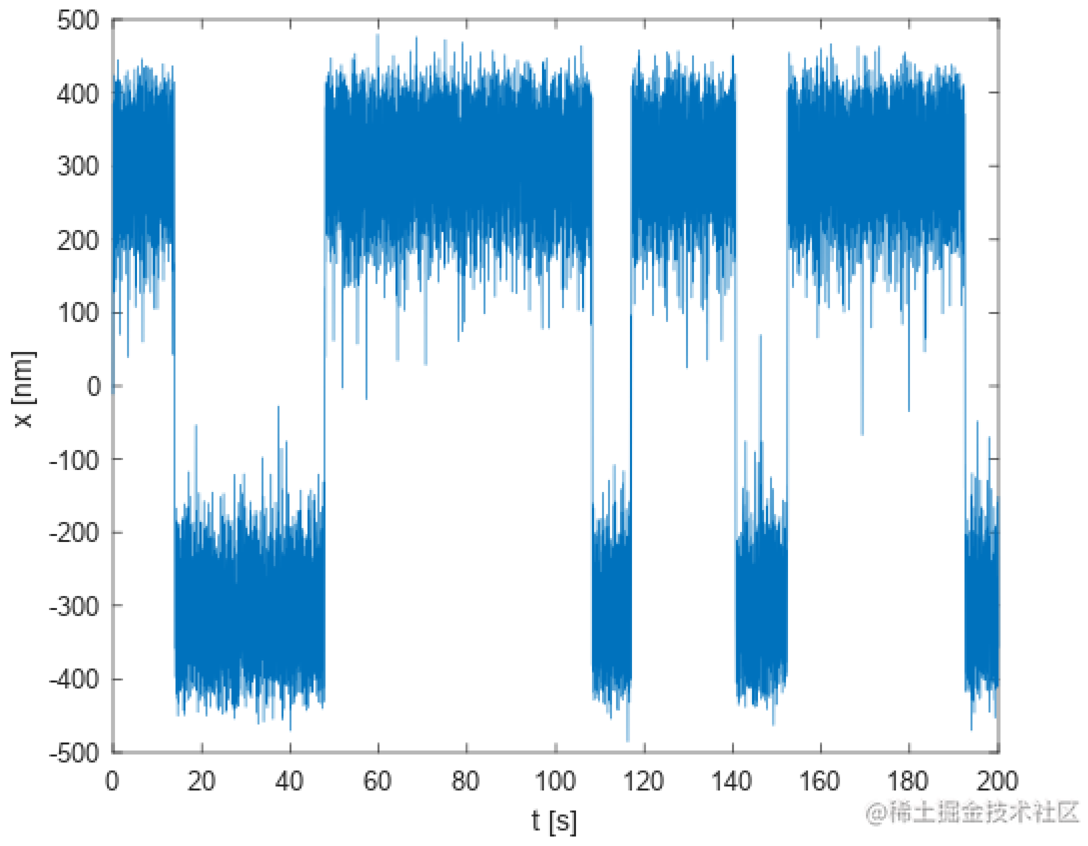
```
function [x,y,t]=...
  forced_trap(N,Dt,x1,y1,R,T,eta,a,b,kx)

  kB = 1.38e-23;  % Boltzmann constant [J/K]
  gamma = 6*pi*R*eta; % friction coefficient
  D = kB*T/gamma; % diffusion coefficient

  %T = 300;          % absolute temprature [K]
  %gamma = 0.001;  % friction coefficient [kg/s]
  m = 1e-15;        % particle mass [kg]
  nm = 1e-9;
  ms = 1e-3;

  c = sqrt(2 * kB * T * gamma);
  f = 1 / (2 * pi) * sqrt(kB * T * gamma / m);

  x(1)=x1;  % initial condition
  y(1)=y1;
  k = kx;
  omega = 132.6;
  t = [0:Dt:(N-1)*Dt];


  for i = 2:1:N
    % Deterministic step
    x(i)  = x(i-1) - (k*x(i-1)+gamma*omega*y(i-1))*Dt/gamma;
    y(i)  = y(i-1) - (-gamma*omega*x(i-1) + k*y(i-1))*Dt/gamma;

    % Diffusive step
    x(i) = x(i) + sqrt(2*D*Dt)*randn();
    y(i) = y(i) + sqrt(2*D*Dt)*randn();
  end

  figure
  plot(x/nm, y/nm)
  xlabel('x [nm]')
  ylabel('y [nm]')
```

```
function [r,s,maxProbability,max_xy]=pacf(x,y,Dt)
r = xcorr(x,ceil(sqrt(length(x))),'unbiased');
maxProbability = max(abs(r));
% normalize the velocity autocorrelation function
r = r / r(ceil(end/2));
% define the delay time length
num_lags = length(r) - 1;
s = Dt*(-num_lags/2:1:num_lags/2);
% s = Dt*[0:1:length(r)-1];
tau = 0.6e-6;
ms = 1e-3;
nm = 1e-9;

figure
plot(s/ms,r,'-black', 'Linewidth', 2.5)
xlabel('$\mathrm{t [ms]}$', 'Interpreter', 'latex', 'FontSize', 14)
ylabel('$C_x(t), C_{xy}(t)$ [n.u.]', 'Interpreter', 'latex', 'FontSize', 14)
hold on;


% Compute the position cross-correlation function
[correlation, lag] = xcorr(x, y,'unbiased');
ms = 1e-3;

% Plot the position cross-correlation function
% Normalization
correlation_normalized = correlation / maxProbability;
max_xy = max(abs(correlation));

num_lags = length(correlation) - 1;
s = Dt*(-num_lags/2:1:num_lags/2);

% Plot the normalized cross-correlation function
plot(s/ms, correlation_normalized,'-','Color',[0.5,0.5,0.5,0.3], 'Linewidth',
2.5);

xlim([-200,200])
ylim([-1,1.1])
```

5.4 Double-well potential force

Consider a double-well potential $U(x) = ax^4/4 - bx^2/2$, then the corresponding force is derived as:

$$F(x) = -ax^3 + bx$$



```
function [x,y,t]=...
  forced_trap(N,Dt,x1,y1,R,T,eta,a,b,kx)

  kB = 1.38e-23;  % Boltzmann constant [J/K]
  gamma = 6*pi*R*eta; % friction coefficient
  D = kB*T/gamma; % diffusion coefficient

  %T = 300;        % absolute temprature [K]
  %gamma = 0.001;  % friction coefficient [kg/s]
  m = 1e-15;      % particle mass [kg]
  nm = 1e-9;
  ms = 1e-3;

  c = sqrt(2 * kB * T * gamma);
  f = 1 / (2 * pi) * sqrt(kB * T * gamma / m);

  x(1)=x1;  % initial condition
  y(1)=y1;
  k = kx;
  omega = 132.6;
  t = [0:Dt:(N-1)*Dt];


  for i = 2:1:N
      % External forces
    F_xDt = (-a * x(i - 1)^3 + b * x(i - 1)) * Dt;

    % Deterministic step
    x(i) = x(i-1) + F_xDt/gamma;
    % Diffusive step
    x(i) = x(i) + sqrt(2*D*Dt)*randn();
  end

  figure
  plot(t,x/nm)
  xlabel('t [s]')
  ylabel('x [nm]')
  xlim([0,200])
```

5.5 Stochastic resonant damping

By introducing the time varying potential, we have stochastic resonant damping:

$$F(x(t),t) = -k[x(t) - x_c sin(2\pi f t)] \tag{22}$$

5.6 Stochastic resonance

and stochastic resonance:

$$F(x(t),t) = -ax^3 + bx + csin(2\pi f t) \tag{23}$$

6. Discussion and Conclusion

After obtaining a numerical solution for the white noise term in the stochastic differential equation using the finite difference method, this paper investigates the particle's motion in two scenarios: free diffusion and within an optical trap.

In the case of free diffusion, at short time scales, the inertial regime equation better describes the particle trajectory than the diffusion regime equation. At long time scales, the results of both equations converge. In the short time scale, the velocity of the particle in the diffusion regime is uncorrelated, and the mean square displacement remains linear at long time scales. However, in the presence of inertia, the particle's mean square displacement undergoes a transition.

Within an optical trap, the particle's motion is significantly influenced by the trap stiffness. A smaller trap stiffness leads to a sparser particle distribution, while a larger trap stiffness results in a denser distribution. In contrast to free diffusion, the mean square displacement does not increase indefinitely in an optical trap but reaches a plateau. The stronger the trap stiffness, the faster the plateau is reached.

Furthermore, for more complex force scenarios, the methods discussed in this paper provide a theoretical basis for specific experimental measurements.

The reproduced simulation closely resembled the original findings, indicating successful replication. However, some minor typo errors are worth noting. Although these do not impact the accuracy of the content in the article, they may cause confusion for readers attempting to reproduce the study. For instance, there is a missing "(dt)^(3/2)" in the inertial regime code in the supplementary file, and there is a discrepancy in the units before and after "kx" in the article. Additionally, there are discussions about some content in the article that might be typo errors, such as the issue with the negative signs in formulas (15) and (18), and whether the equation for the particle's position autocorrelation function in Figure 5(a) is indeed formula (23).

The reproduction of the simulation of a Brownian particle in an optical trap demonstrates the robustness of the original study's methodology. Despite some challenges, the reproduced results largely align with the original findings, reaffirming the validity of the proposed model.

7. Future Work

Future efforts may focus on fine-tuning simulation parameters to better match the original conditions. Additionally, exploring alternative simulation techniques or software platforms could provide further insights and validation. The Python code refactoring for all MATLAB code in the literature will continue, and updates will be subsequently provided in the GitHub repository.

> [1] Giorgio Volpe, Giovanni Volpe. Simulation of a Brownian particle in an optical trap, Am. J. Phys. 81, 224–230 (2013).