Stochastic simulation for an active Brownian particle

1. **Introduction**

In this project, a self-propelled particle modelled by the stochastic differential equations (1) to (3) will be studied:

|  |  |
| --- | --- |
|  | Here is the active constant velocity,  is the position,  is the direction of motion,  is the rotational diffusion coefficient, and  is a Gaussian white noise satisfying and |

The autocorrelation of the direction of motion and the long term mean square displacement will be focused.

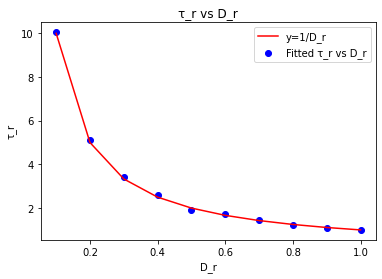
1. **Dynamics**

The system of stochastic differential equations (1) to (3) models a particle which propels itself with speed while undergoing Brownian motion in orientation. In other words, its orientation undergoes free diffusion such that the direction of motion are random. This type of dynamics is referred to as rotational diffusion dynamics.

1. **Autocorrelation of the direction of motion**

The autocorrelation of the direction of motion is given by:

The autocorrelation is an exponential decay function in time, which approaches 0 when *t* is getting more positive or negative. It means that the dependence of the direction of motion to its lag-t direction decreases as t increases. The speed of decay is governed by which depends on the rotational diffusion coefficient through the relation . The smaller the , the faster the decay. The dependence of on could be simulated with the code in part (A) of the Appendix and the relation could be visualized as figure below.



1. **Mean square displacement (MSD)**

The mean square displacement measures how far on average the particle is from the initial position. For the active Brownian particle described by equations (1) to (3), the theoretical MSD is given by:

For , , which is a linear function of *t*. It means that the mean square displacement will grow linearly with rate when . Such theoretical result could be verified through simulation with the code in part (B) of the Appendix and the result could be visualized as figures below.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

1. **Appendix (source code)**
2. **Dependence of on**

import numpy as np

from numpy import sin, cos, exp, sqrt

import matplotlib.pyplot as plt

from scipy.optimize import curve\_fit

np.random.seed(5003)

samples = 50

N = 1000  # number of steps

h = 0.1 # step size

Ts = np.arange(0, N\*h, step=h)

phi = np.zeros(N) # array for direction of motion

Dr = np.linspace(0.1,1,10)

tau\_r = np.zeros(10)

z\_mu = 0

z\_sd = sqrt(1/h)

def auto\_corr(t, tau):

    return exp(-abs(t)/tau)

for idx, dr in enumerate(Dr):

    print(f'Dr = {dr:.1f}')

    acf = np.zeros((samples, N))

    for ns in range(samples):

        zetas = np.random.normal(z\_mu, z\_sd, (N-1))

        # Euler method for phi, x and y

        for i in range(1,N):

            phi[i] = phi[i-1] + h\*zetas[i-1]\*sqrt(2\*dr)

        sinphi = sin(phi)

        cosphi = cos(phi)

        for t in range(N):

            ac = 0

            for s in range(N-t):

                ac += cosphi[s]\*cosphi[s+t]+sinphi[s]\*sinphi[s+t]

            acf[ns,t] = ac/(N-t)

    tau0 = 1

    macf = acf.mean(axis=0)

    c = int(1/(dr\*h))

    tau\_fit = curve\_fit(auto\_corr, Ts[:c], macf[:c], p0=[tau0])[0][0]

    print(f'tau\_r = {tau\_fit:.2f}')

    tau\_r[idx] = tau\_fit

plt.scatter(Dr, tau\_r, c='b', label='Fitted \u03C4\_r vs D\_r')

plt.plot(Dr, 1/Dr, c='r', label='y=1/D\_r')

plt.xlabel('D\_r')

plt.ylabel('\u03C4\_r')

plt.title('\u03C4\_r vs D\_r')

plt.legend()

plt.show()

1. **MSD with selected and**

np.random.seed(5003)

samples = 3000

N = 1000  # number of steps

h = 0.1 # step size

Ts = np.arange(0, N\*h, step=h)

phi = np.zeros(N) # array for direction of motion

x = np.zeros(N) # array for x

y = np.zeros(N) # array for y

Dr = [0.3,0.5,0.7]

V = [0.1,0.3,0.5,0.7,0.9]

msd = np.zeros((10,N))

z\_mu = 0

z\_sd = sqrt(1/h)

for v in V:

    plt.figure(figsize=(15,5))

    for idx, dr in enumerate(Dr):

        sd = np.zeros((samples, N))

        for ns in range(samples):

            zetas = np.random.normal(z\_mu, z\_sd, (N-1))

            for i in range(1,N):

                phi[i] = phi[i-1] + h\*zetas[i-1]\*sqrt(2\*dr)

                x[i] = x[i-1] + h\*v\*cos(phi[i-1])

                y[i] = y[i-1] + h\*v\*sin(phi[i-1])

                sd[ns,i] = x[i]\*\*2 + y[i]\*\*2

        # Plotting

        plt.subplot(1,len(Dr),idx+1)

        plt.plot(Ts, sd.mean(axis=0), 'b', label='Simulated MSD')

        plt.plot(Ts, 2\*v\*\*2\*Ts/dr, 'r', label=f'y=2v^2\u03C4\_rt')

        plt.title(f'MSD(t) for v={v:.2f} and \u03C4\_r={1/dr:.1f}')

        plt.xlabel('Time')

        plt.ylabel('MSD')

        plt.legend()

    plt.show()