# Python codes

A. L. Merino-Díaz, E. Ramírez-Solano, R. Ramírez-Sánchez

October 5, 2021

**Abstract**

For a better understanding of the orientational behavior, we provide the Python codes to recreate the simulations presented in the article. In each section we present the code according to the number of Figure. For this article we use Python 2.7, the libraries use here can be different from the users.

## 1 Figure 1

```
from visual import
from visual.graph import
import matplotlib.pyplot as plt
import numpy as np

   ##Initial values

N=300; # number of samples
pa=0;
pb=0;
pc=0;
palist=[]
pblist=[]
pclist=[]

   ##Equation 4:

dta=1; #timestep (a)
stepa=N*dta;
wa=(1/sqrt(dta))*np.random.normal(0,1,N)
ta=(np.arange(0,stepa,dta))

dtb=0.5; #timestep (b)
stepb=N*dtb;
wb=(1/sqrt(dtb))*np.random.normal(0,1,N)
tb=(np.arange(0,stepb,dtb))

dtc=0.1; #timestep (c)
stepsc=N*dtc;
wc=(1/sqrt(dtc))*np.random.normal(0,1,N)
tc=(np.arange(0,stepc,dtc))

   ##Equation 7:

for i in range(N):
pa=pa+sqrt(dta)*random.normal(0,1)
palist.append(pa)
```

```
pb=pb+sqrt(dtb)*random.normal(0,1)
pblist.append(pb)
pc=pc+sqrt(dtc)*random.normal(0,1)
pclist.append(pc)

   ##Data presentation

plt.figure()
plt.subplot(2,3,1)
plt.plot(ta,wa,'ro')
plt.ylabel('$W_i$')

plt.subplot(2,3,2)
plt.plot(tb,wb,'ro')

plt.subplot(2,3,3)
plt.plot(tc,wc,'ro')

plt.subplot(2,3,4)
plt.plot(ta,palist)
plt.xlabel('time')
plt.ylabel('$theta_i$')

plt.subplot(2,3,5)
plt.plot(tb,pblist)
plt.xlabel('time')

plt.subplot(2,3,6)
plt.plot(tc,pclist)
plt.xlabel('time')

plt.show()
```

## 2 Figure 2

```
## Initial values

Nc=30; #Samples at short times
Nl=5000; #Samples at long times
dt=1e-8; #time step

thetac=[0 for i in range(Nc)] #initial inertial angular position
thetapc=[0 for i in range(Nc)] #initial non inertial angular position
thetal=[0 for i in range(Nl)]
thetapl=[0 for i in range(Nl)]
vlist=[];
vplist=[];

R=1e-6; #particle size
T=300.0; #system temperature
eta=0.001; #viscosity
d=2.6e3; #density

kB=1.38e-23; #thermal energy
gamma=8.0*pi*eta*(R**3); #friction coefficient
m=(4.0/3.0)*pi*(R**3)*d; #mass
```

```
Dr=kB*T/gamma; #rotational diffusion coefficient
I=(2.0/3.0)*m*(R**2); #moment of inertia
tau=I/gamma; #relaxation time


   ## Equation 13

for i in range(2,Nc):
z=random.normal(0,1)
thetac[i]=((2.0+dt*gamma/I)/(1+dt*gamma/I))*thetac[i-1]-(1/(1+dt*gamma/I))*thetac[i-2]+...
           (sqrt(2.0*kB*T*gamma)/(I+dt*gamma))*((dt)**(3.0/2.0))*z
thetapc[i]=thetapc[i-1]+sqrt(2.0*Dr*dt)*z
tc=[i*dt/tau for i in range(Nc)]

for i in range(2,Nl):
z=random.normal(0,1)
thetal[i]=((2.0+dt*gamma/I)/(1+dt*gamma/I))*thetal[i-1]-(1/(1+dt*gamma/I))*thetal[i-2]+...
           (sqrt(2.0*kB*T*gamma)/(I+dt*gamma))*((dt)**(3.0/2.0))*z
thetapl[i]=thetapl[i-1]+sqrt(2.0*Dr*dt)*z
tl=[i*dt/tau for i in range(Nl)]


   ## Equation 15

for i in range(Nc-1):
v=(thetac[i+1]-thetac[i])/(dt/tau)
vlist.append(v)
vp=(thetapc[i+1]-thetapc[i])/(dt/tau)
vplist.append(vp)

r = np.correlate(vlist,vlist,mode='full')
rp= np.correlate(vplist,vplist,mode='full')
s=[i*dt/tau for i in range(-len(r)/2,len(r)/2)]


   ##Data presentation


plt.figure()
plt.subplot(1,3,1)
plt.plot(tc,thetac,tc,thetapc,'r--')
plt.xlabel('$t/ tau$')
plt.ylabel('$  theta_i$ [rad]')

plt.subplot(1,3,2)
plt.plot(tl,thetal,tl,thetapl,'r--')

plt.subplot(1,3,3)
plt.plot(s,r,s,rp,'r--')
plt.xlabel('$t/ tau$') plt.ylabel('$C_{ Omega}$(t)[u.a.]')
plt.show()
```