Homework 4

Homework 4-1

0.0/1.0 point (graded)

Eq.(F11) represents the condition that the random forces $\Delta \mathbf{W}_i = (\Delta W_{i,x}, \Delta W_{i,y}, \Delta W_{i,z})$ should satisfy in the case of a Brownian Motion with uncorrelated forces.

$$\langle \Delta \mathbf{W}_i \Delta \mathbf{W}_j \rangle = 2k_B T \zeta \Delta t \mathbf{I} \delta_{ij} \tag{F11}$$

Assuming that Eq.(F11) is satisfied, which of the following equations are also valid (i.e., which of the following are also satisfied)?

$$\langle \Delta \mathbf{W}_i \cdot \Delta \mathbf{W}_j \rangle = 2k_B T \zeta \Delta t \delta_{ij} \tag{FH11}$$

$$\langle \Delta \mathbf{W}_i \cdot \Delta \mathbf{W}_j \rangle = 6k_B T \zeta \Delta t \delta_{ij} \tag{FH12}$$

$$\langle \Delta \mathbf{W}_i \Delta \mathbf{W}_j \rangle = 2k_B T \zeta \Delta t \delta_{i,j} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \tag{FH13}$$

$$\langle \Delta W_{i,\alpha} \Delta W_{j,\beta} \rangle = 6k_B T \zeta \Delta t \delta_{\alpha\beta} \delta_{ij} \qquad (\alpha, \beta \in x, y, z) \tag{FH14}$$

- Eq.(FH11) only
- Eq.(FH12) only
- Eq.(FH13) only
- Eq.(FH14) only

Submit You have used 0 of 2 attempts

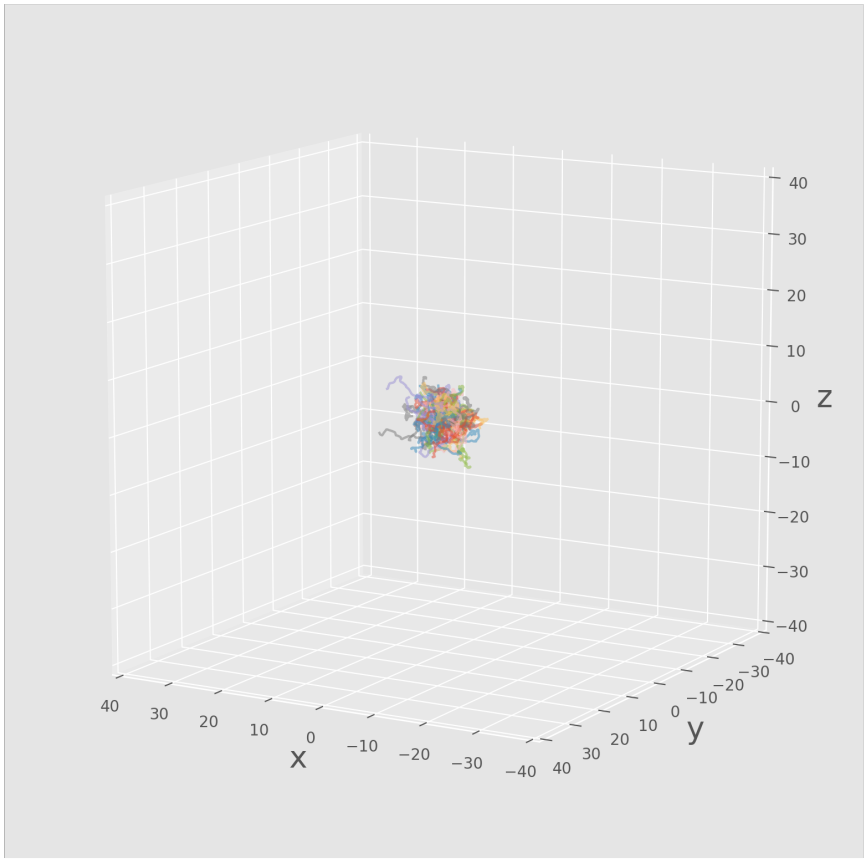
Homework 4-2

0.0/1.0 point (graded)

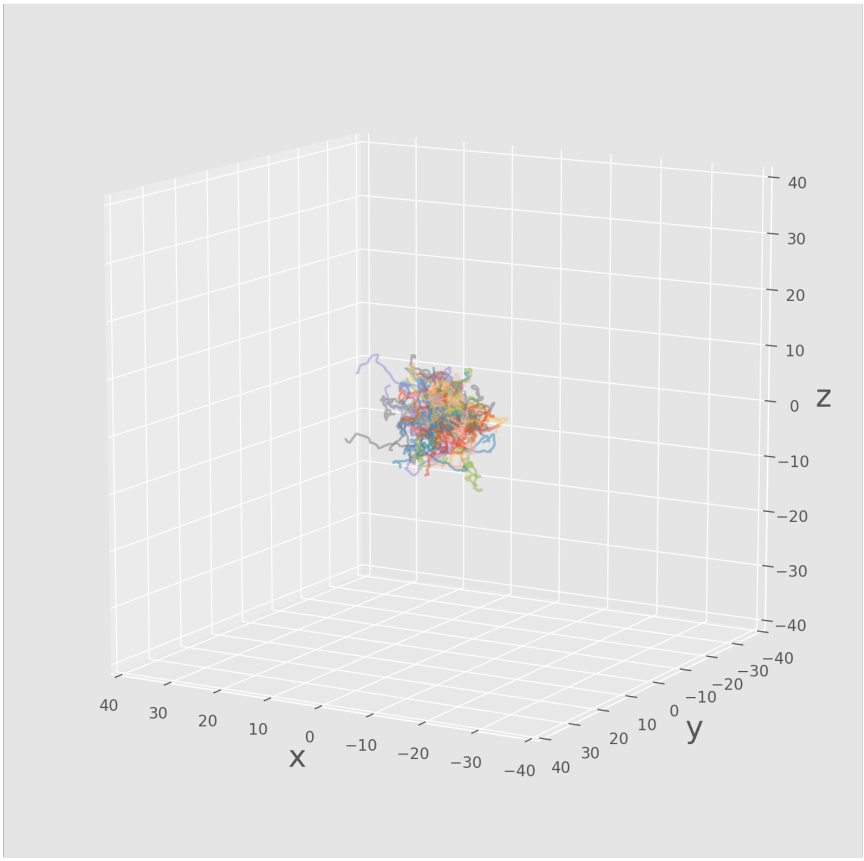
Perform the same simulation presented in the video, using the original code example introduced in Part 2, but this time change the total number of steps and step size to be nums $= 2048$ and $\Delta t = 0.025$, respectively.

Plot the trajectories of the particles in 3D space. Which of the following graphs (G21 - G25) is the closest to what you obtained?

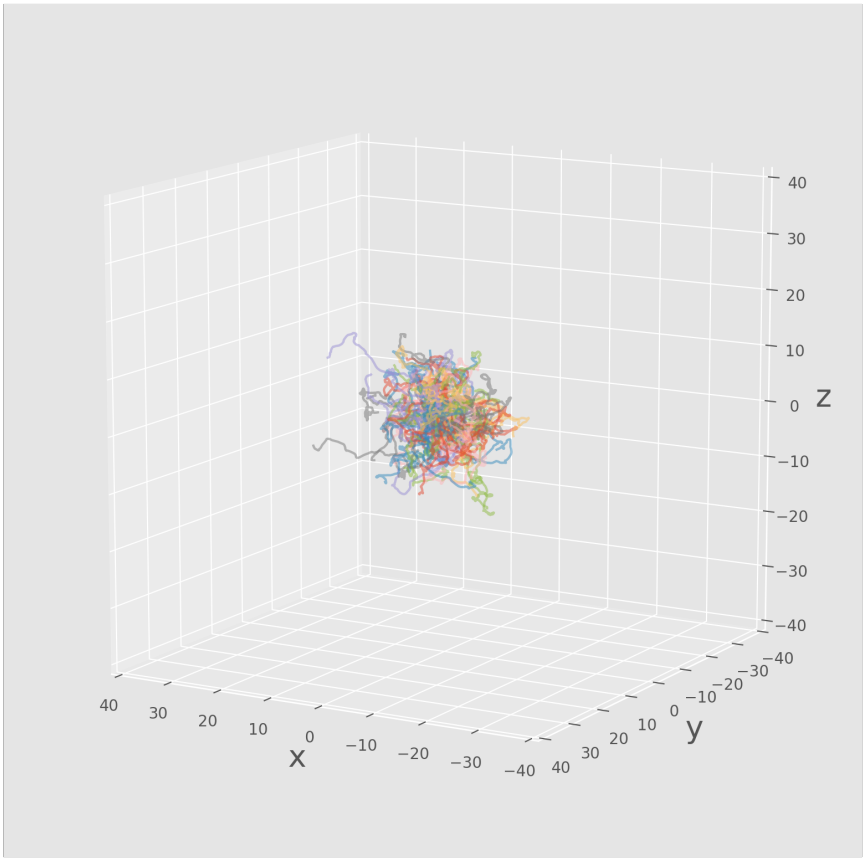
G21



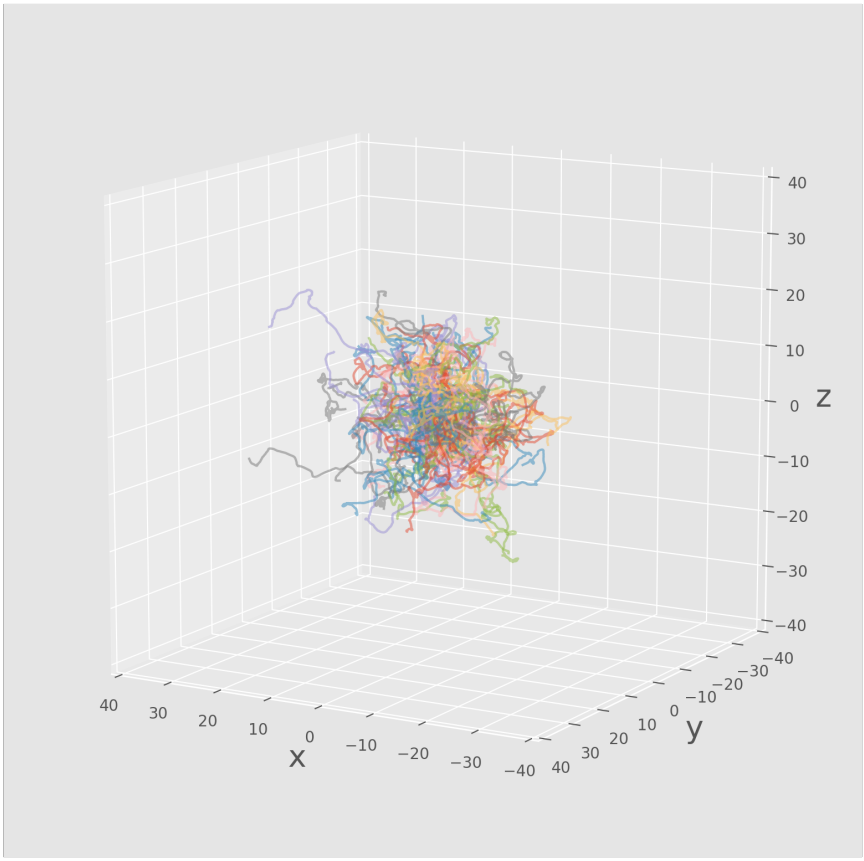
G22



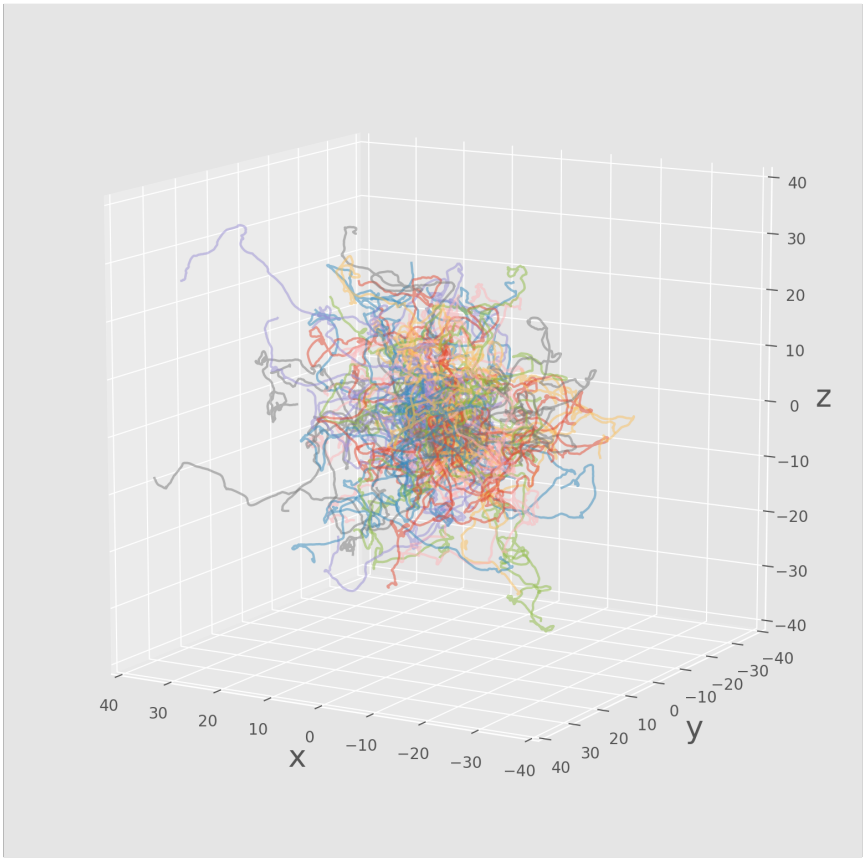
G23



G24



G25



- G21
- G22
- G23
- G24
- G25

Submit

You have used 0 of 2 attempts

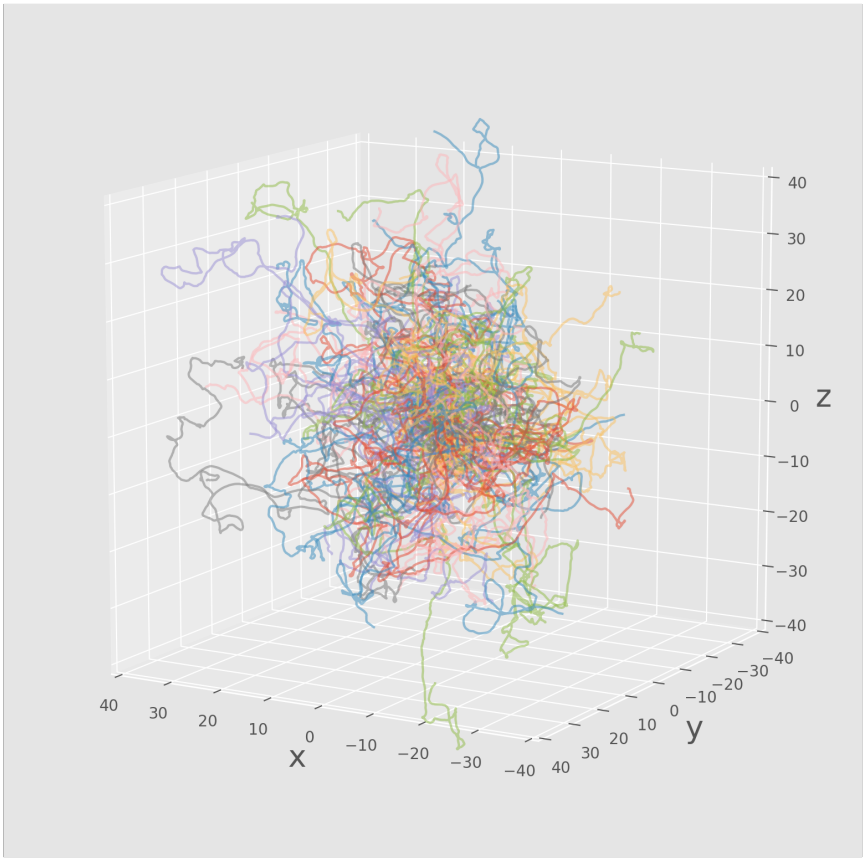
Homework 4-3

0.0/1.0 point (graded)

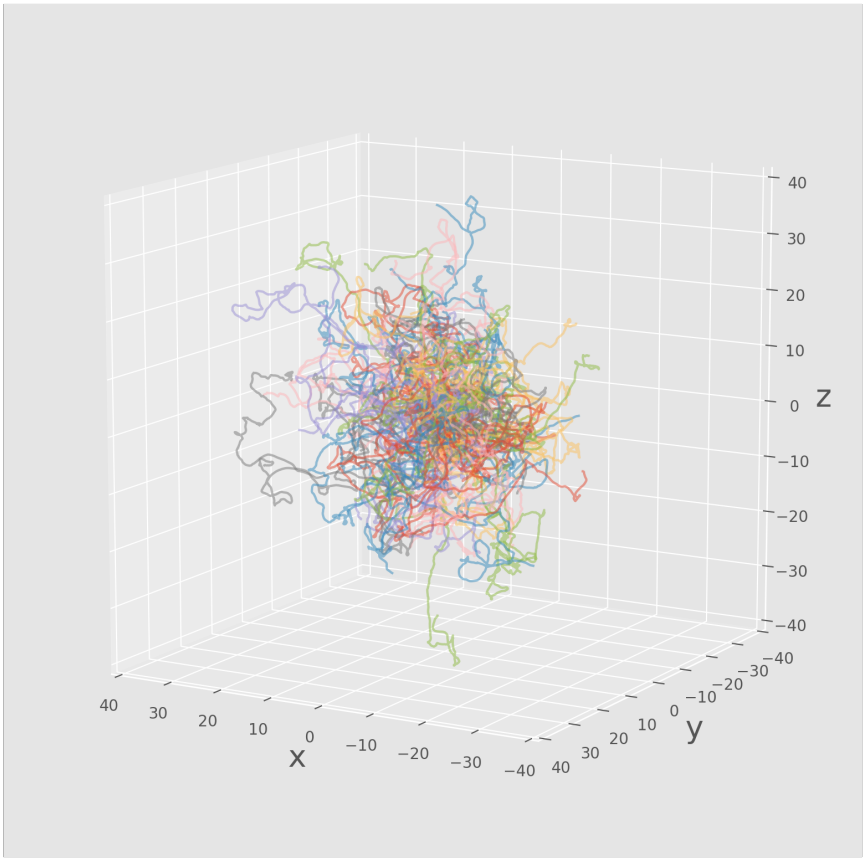
Perform the same simulation presented in the video, using the original code example introduced in Part 2, but now changing the number of steps and step size to be nums $= 4096$ and $\Delta t = 0.0125$, respectively.

Plot the trajectories of the particles in 3D space. Which of the following graphs (G31 - G35) is the closest to what you obtained?

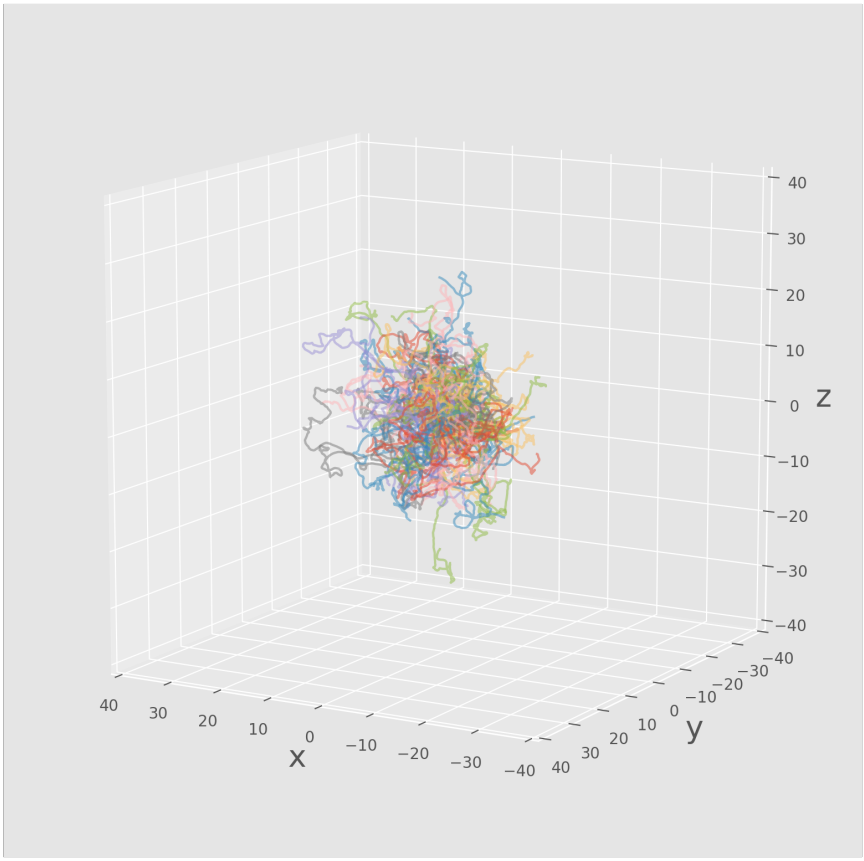
G31



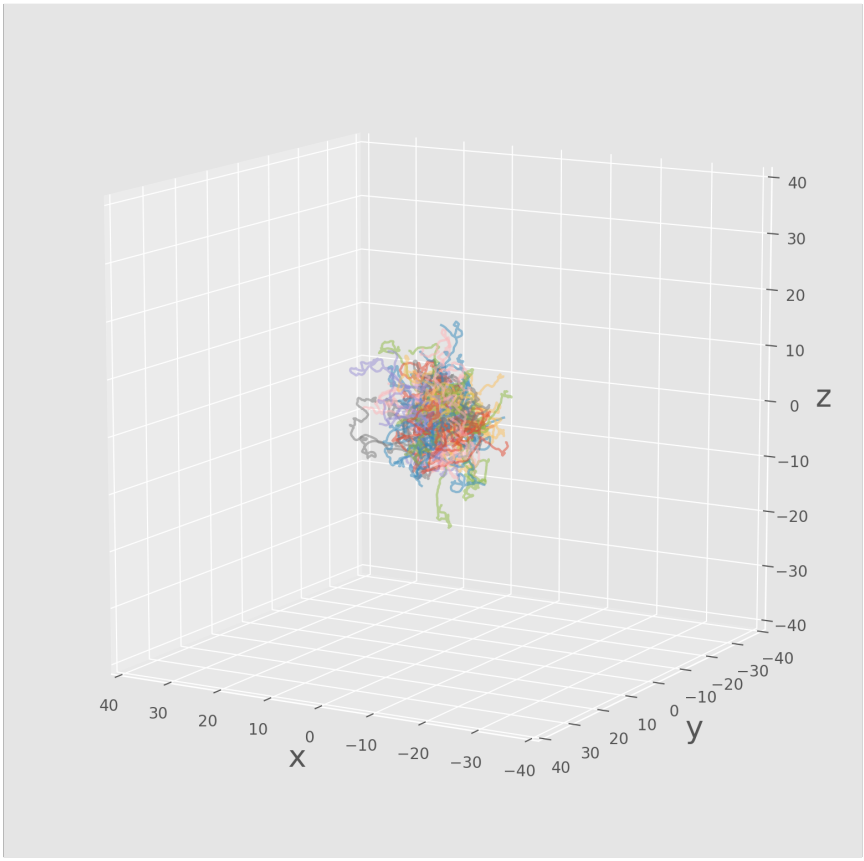
G32



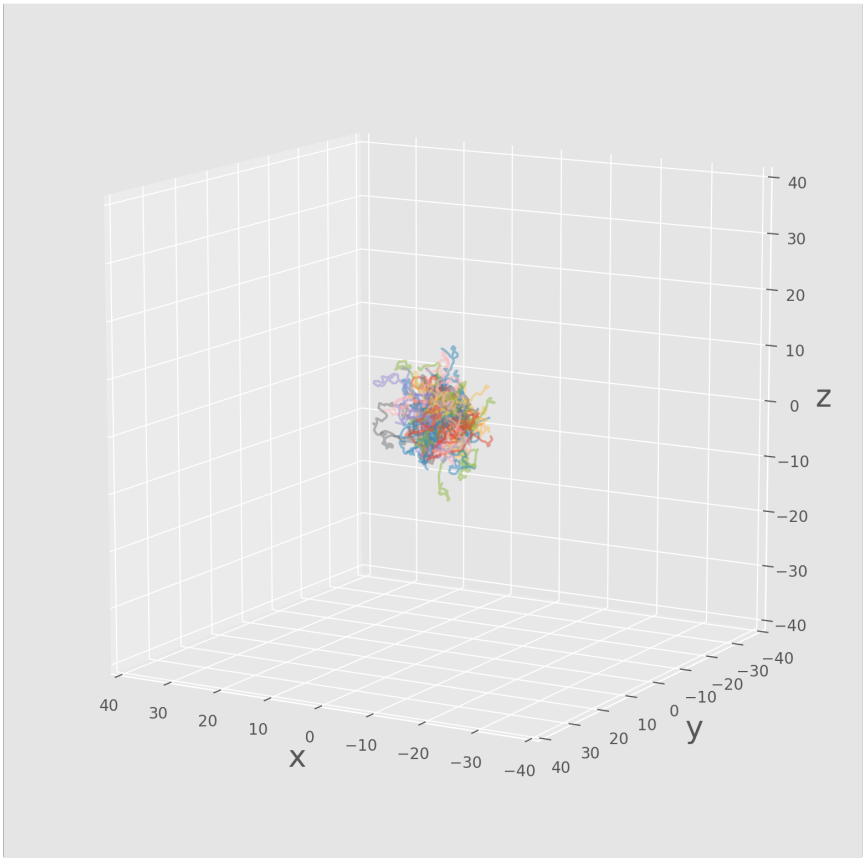
G33



G34



G35



- G31
- G32
- G33
- G34
- G35

Submit

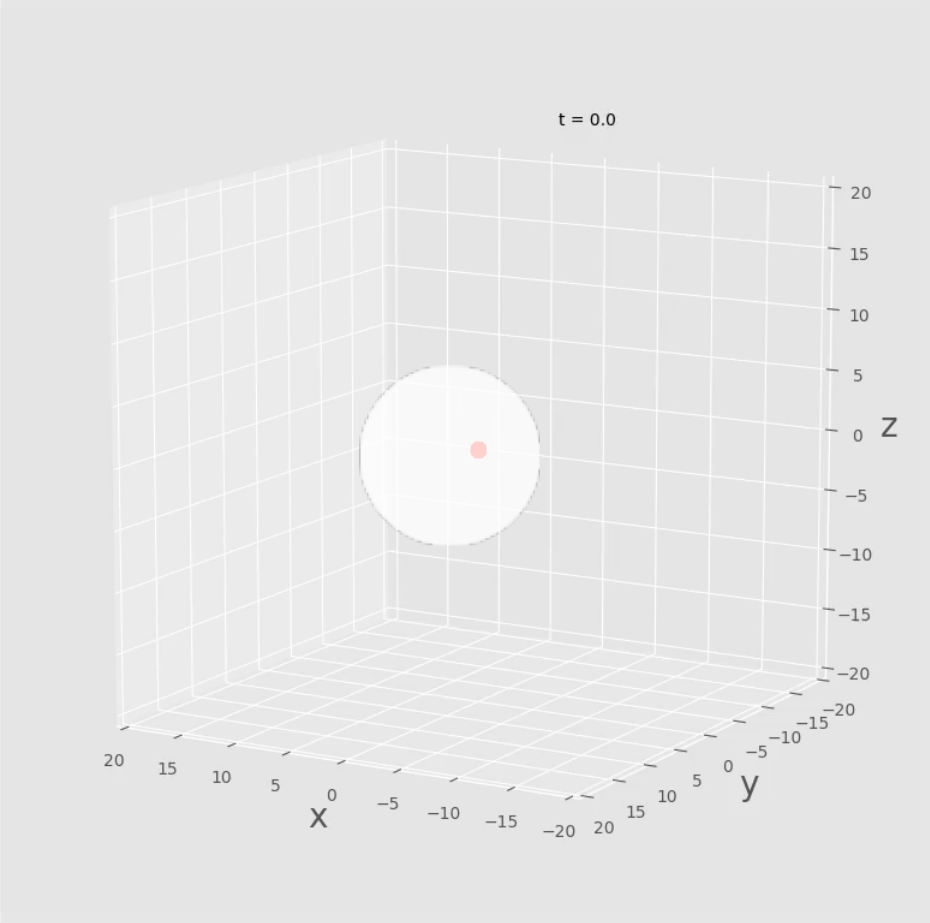
You have used 0 of 2 attempts

Homework 4-4 (Question)

Perform the same simulation presented in the video, using the original code example introduced in Part 3, but changing the friction coefficient, mass, and thermal energy to be $\zeta = 0.5$, $m = 0.5$, and $kBT = 0.5$, respectively.

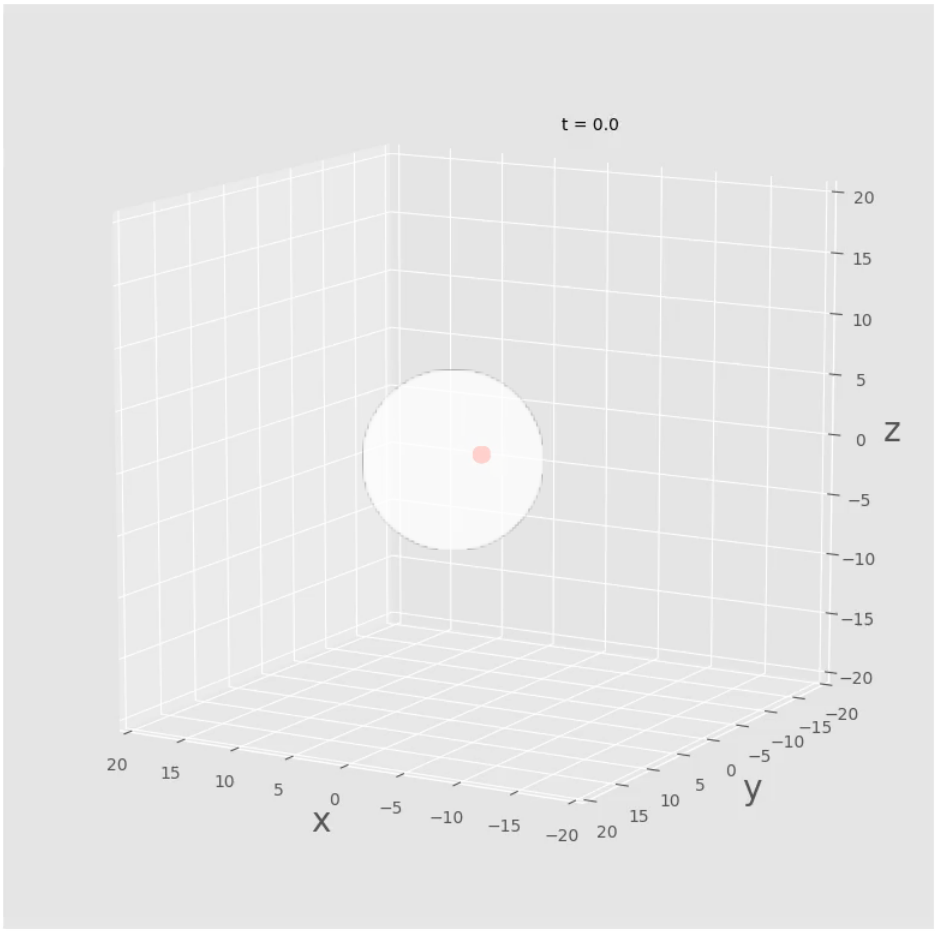
Which of the following animations (A01 - A04) is the closest to what you obtain?

- A01



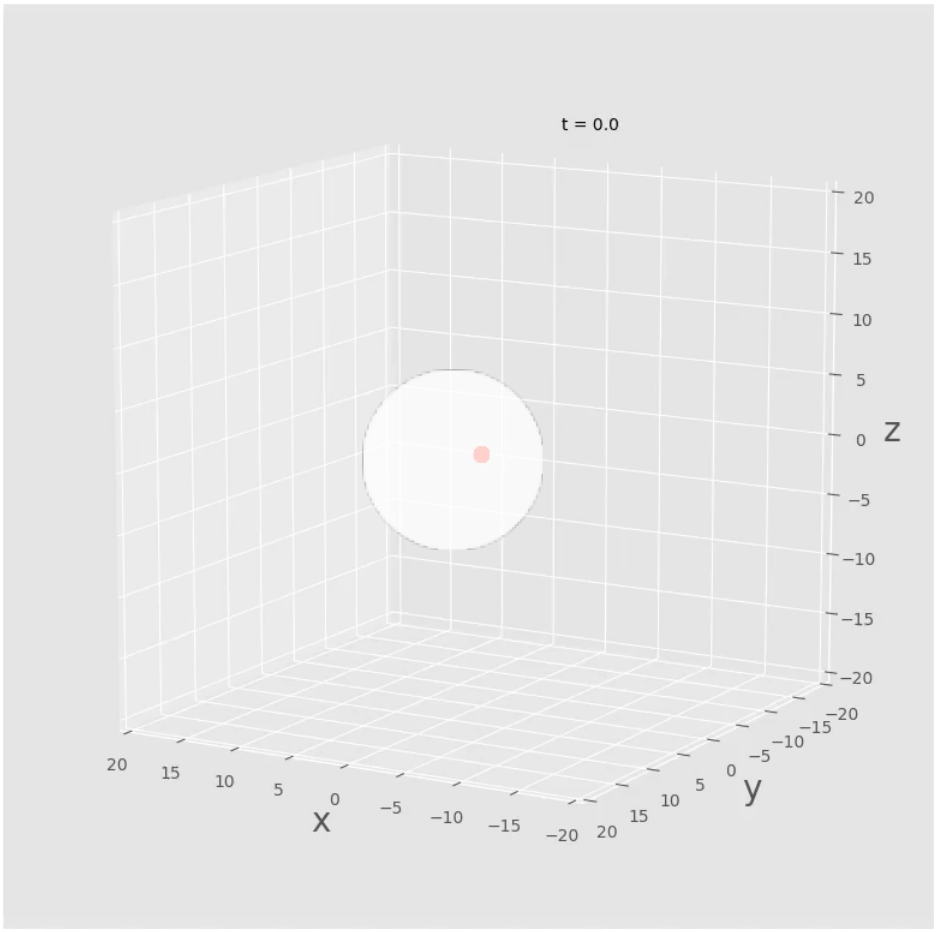
0:00 / 0:20

- A02



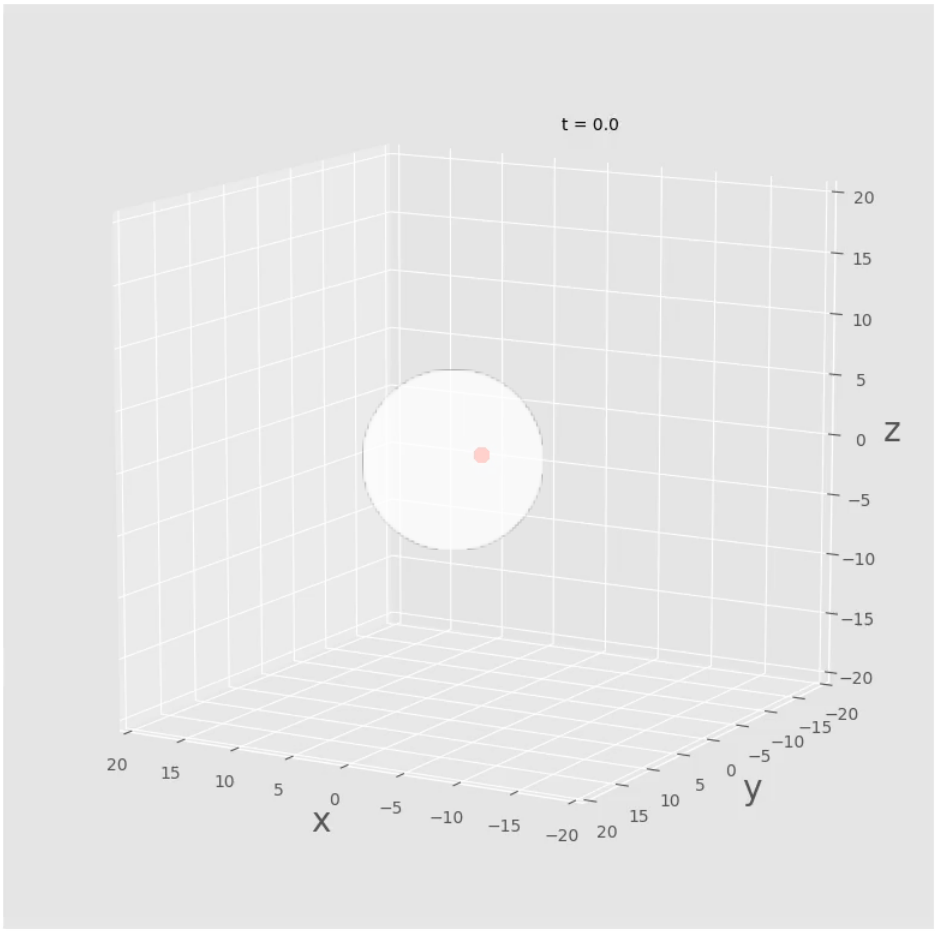
0:00 / 0:20

- A03



0:00 / 0:20

- A04



0:00 / 0:20

Homework 4-4 (answer)

0.0/1.0 point (graded)

- A01
- A02
- A03
- A04

Submit You have used 0 of 2 attempts

Homework 4-5

0.0/2.0 points (graded)

Modify the original code presented in Part 3, which used the Euler method, to solve for the motion of Brownian particles with a 2nd order Runge-Kutta method. This can be achieved through a suitable modification of the following animate function.

```
def animate(i):
    global R,V,W,Rs,Vs,Ws,time
    time[i]=i*dt
    W = std*np.random.randn(nump,dim)
    V = V*(1-zeta/m*dt)+W/m
    R = R+V*dt
    Rs[i,:,:]=R
    Vs[i,:,:]=V
    Ws[i,:,:]=W
    title.set_text(r"t = "+str(time[i]))
    line.set_data(Rs[:i+1,n,0],Rs[:i+1,n,1])
    line.set_3d_properties(Rs[:i+1,n,2])
    particles.set_data(R[:,0],R[:,1])
    particles.set_3d_properties(R[:,2])
   return particles,title,line
```

Choose the most appropriate animate function from the codes shown below (C1, C2, C3, C4).

```
# C1
V1 = np.zeros([nump,dim])
def animate(i):
    global R,V,W,Rs,Vs,Ws,time
    time[i]=i*dt
    W = std*np.random.randn(nump,dim)
    V1 = V*(1-zeta/m*0.5*dt)+W/m
    V = V-V1*zeta/m*dt+W/m
    R = R+V1*dt
    Rs[i,:,:]=R
    Vs[i,:,:]=V
    Ws[i,:,:]=W
    title.set_text(r"t = "+str(time[i]))
    line.set_data(Rs[:i+1,n,0],Rs[:i+1,n,1])
    line.set_3d_properties(Rs[:i+1,n,2])
    particles.set_data(R[:,0],R[:,1])
    particles.set_3d_properties(R[:,2])
    return particles,title,line
```

```
# C2
V1 = np.zeros([nump,dim])
def animate(i):
    global R,V,W,Rs,Vs,Ws,time
    time[i]=i*dt
    W = std*np.random.randn(nump,dim)
    V1 = V*(1-zeta/m*0.5*dt)+W/m/np.sqrt(2)
    V = V1*(1-zeta/m*dt)+W/m
    R = R+V1*dt
    Rs[i,:,:]=R
    Vs[i,:,:]=V
    Ws[i,:,:]=W
    title.set_text(r"t = "+str(time[i]))
    line.set_data(Rs[:i+1,n,0],Rs[:i+1,n,1])
    line.set_3d_properties(Rs[:i+1,n,2])
    particles.set_data(R[:,0],R[:,1])
    particles.set_3d_properties(R[:,2])
    return particles,title,line
```

```
# C3
V1 = np.zeros([nump,dim])
def animate(i):
    global R,V,W,Rs,Vs,Ws,time
    time[i]=i*dt
    W = std*np.random.randn(nump,dim)
    V1 = V*(1-zeta/m*0.5*dt)
    V = V-V1*zeta/m*dt+W/m
    R = R+V1*dt
    Rs[i,:,:]=R
    Vs[i,:,:]=V
    Ws[i,:,:]=W
    title.set_text(r"t = "+str(time[i]))
    line.set_data(Rs[:i+1,n,0],Rs[:i+1,n,1])
    line.set_3d_properties(Rs[:i+1,n,2])
    particles.set_data(R[:,0],R[:,1])
    particles.set_3d_properties(R[:,2])
    return particles,title,line
```

```
# C4
V1 = np.zeros([nump,dim])
def animate(i):
    global R,V,W,Rs,Vs,Ws,time
    time[i]=i*dt
    W = std*np.random.randn(nump,dim)
    V1 = V*(1-zeta/m*0.5*dt)
    V = V1*(1-zeta/m*dt)+W/m
    R = R+V1*dt
    Rs[i,:,:]=R
    Vs[i,:,:]=V
    Ws[i,:,:]=W
    title.set_text(r"t = "+str(time[i]))
    line.set_data(Rs[:i+1,n,0],Rs[:i+1,n,1])
    line.set_3d_properties(Rs[:i+1,n,2])
    particles.set_data(R[:,0],R[:,1])
    particles.set_3d_properties(R[:,2])
    return particles,title,line
```

- C1
- C2
- C3
- C4

Submit You have used 0 of 2 attempts