Project Outline: Simulating 2D Diffusion-Limited Aggregation and Brownian Trees

I aim to model two-dimensional diffusion-limited aggregation (DLA), a process involving the aggregation, or clustering, of particles undergoing Brownian motion. When such particles are allowed to adhere to a seed point, clusters (Brownian trees) are formed with distinct tree-like shapes.

To begin, I intend to model simple random walks, wherein randomly moving particles wander from a starting point. Assuming a constant step size, I will model a series of small, discrete particles in 1D, 2D and finally 3D, using *numpy.random.choice* to choose a unit step in a given direction. I will extend this to variable step size using *numpy.random.randn*: $x(t_{i+1}) = x(t_i) + \sqrt{T/(N-1)}\,\mathcal{N}(0,1)$ where $x(t)$is displacement, T is total time, N is no. of steps, and $\mathcal{N}(0,1)$ is the standard normal distribution. In all dimensions, the average displacement $\langle d \rangle$ after N steps should be zero, since there is an equal probability of travelling in any direction. The rms displacement $\sqrt{\langle d \rangle^2}$ should be $\sqrt{N}$. I will test these probabilistic characteristics to verify this simple model is as random as possible, and hence not biased.

Since random walks exist in discrete time and space, taking a smaller time step will allow convergence towards Brownian motion, a time continuous stochastic (random) process. I will hence create a class using *scipy.integrate.odeint* to implement and solve the Langevin equation: $m\frac{d\boldsymbol{v}}{dt} = -\xi\boldsymbol{v} + \boldsymbol{\eta}(t)$, where m is mass, $\boldsymbol{v}$ is velocity, ξ is the friction coefficient given by Stokes' law, and $\boldsymbol{\eta}(t)$ is a 'noise' term arising due to randomness and representing collisions. This equation describes the time evolution and collisions arising from Brownian motion, and contains both frictional and random forces.

Finally, I will inherit the relevant properties of this Brownian motion class, and extend to DLA by introducing a 'seed' particle. New particles will be generated at the system boundaries, and undergo Brownian motion until they are close enough to 'stick' to the seed/cluster. The 'stickiness' of each particle is analogous to its electrostatic force, which I will change to vary interaction strength. I may verify this system by first implementing a 2D pixel grid, emulating a lattice with discretised positions.

I will explore the variation of different parameters, such as:

- number and size mixes of particles ('walkers')
- step size and consistency (uniform/variable)
- system boundaries (size, bounded/unbounded)
- 'stickiness/ sticking coefficient' (changes density of tree)
- 'attractor geometries' (seed point, line, circle, any other shape)

In addition to the regular Python modules (e.g. *numpy, matplotlib, scipy*), I intend to use:

- *random*/*numpy.random.randn* - generate uniform/normally dist. pseudo-random numbers
- *numpy.random.choice* - randomly choose given values
- *scipy.integrate.odeint* - solve stochastic differential equations (Langevin, maybe others)
- *pandas* - store and save data in dataframes, should improve code efficiency
- *matplotlib.animation* - create simple animations
- *pygame* - host the application for running the final DLA simulation
- *pytest* - test functions and classes

If time allows, I will extend this simulation to 3D (including visualisations), and analyse the corresponding physical properties/change in parameters accordingly. External forces (e.g. potentials) could additionally be applied to introduce bias in 'walker' direction.