**Q1:**

**Outline:**

In this question, we explore Brownian motion and try to simulate a random walk model in python.

We achieve this through the random number generator in python that uses the Mersenne twister algorithm. This will be done through assigning directions to specific integers and then generating random integers and use it to guide the particle. Boundary conditions also apply.

In addition to this, another model, namely the DLA (Diffusion Limited Aggregation). In this model, particles stick to the boundaries as well as each other and as they do, new particles are introduced at the center of the grid. This will produce a tree branck-like structure that will extend to the center of the grid and prevent more particles from starting off.

**Results:**

**Part-a:**

After 5000 iterations of the random walk model, we observe diffusive patterns similar to the ones seen in diffusing ink in water. The diagram for a single particle is as following:

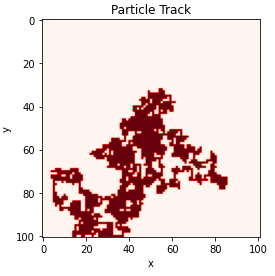


Figure-1 : Track of a Brownian Particle over 5000 iterations of a single particle. Dark spots demonstrate points the particle crossed.

**Part-b:**

Now we implement the DLA algorithm for a 101 by 101 grid. We expect to have a branch like structure after the algorithm finishes running. The yellow spots will represent the particles on a dark background. We both analyze the result for the first 100 stuck particles as well as the final result:

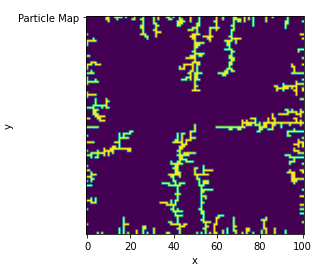


Figure-2: The DLA pattern for 101x101 grid. Number of stuck particles is 1040.

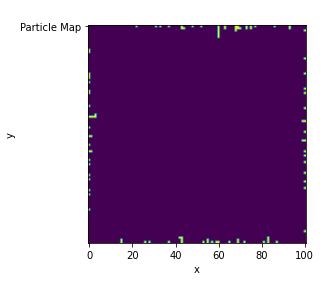


Figure-3: The DLA pattern for 101x101 grid. Number of stuck particles is exactly 100 (first 100 of the bunch as figure-2).

**Part-c:**

This is the pattern generated for a 201x201 grid. The bright pixcels represent particles.

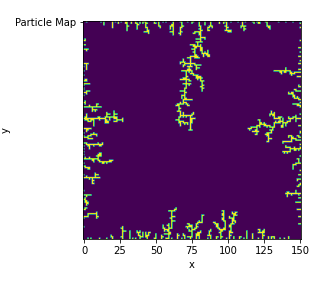


Figure-4: A completed DLA pattern for a 151x151 grid. Particles are shown with bright colors.

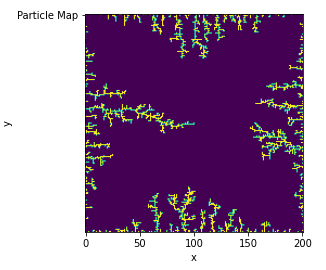


Figure-5: A completed DLA pattern for a 201x201 grid. Particles are shown with bright colors.

**Q2:**

**Outline:**

in this question, we explore the idea of the mean value Monte Carlo method for integration. We will demonstrate the volume integral of n-dimensional spheres of radius one through this concept.

In this question, we pick n = 10.

Through the hit and miss process, we generate a million random points bounded to a 10 dimensional cube and record the ones that hit the inside of the hyper-sphere.

**Results:**

Our program returns:



We calculate the integral and its error for the Mean value Monte Carlo integration through the following expressions:

=

Where I stands for the value of the integral and **σ** for the error associated to it.

We calculated the variance of function ***f*** through the variance function in numpy applied to our array of values of the function ***f***.

Note that this function returns ones and zeros only. (one within the sphere and zero outside)

Our Calculated value is very close to the actualy value that can be obtained through the explicit formula in the lab manual.