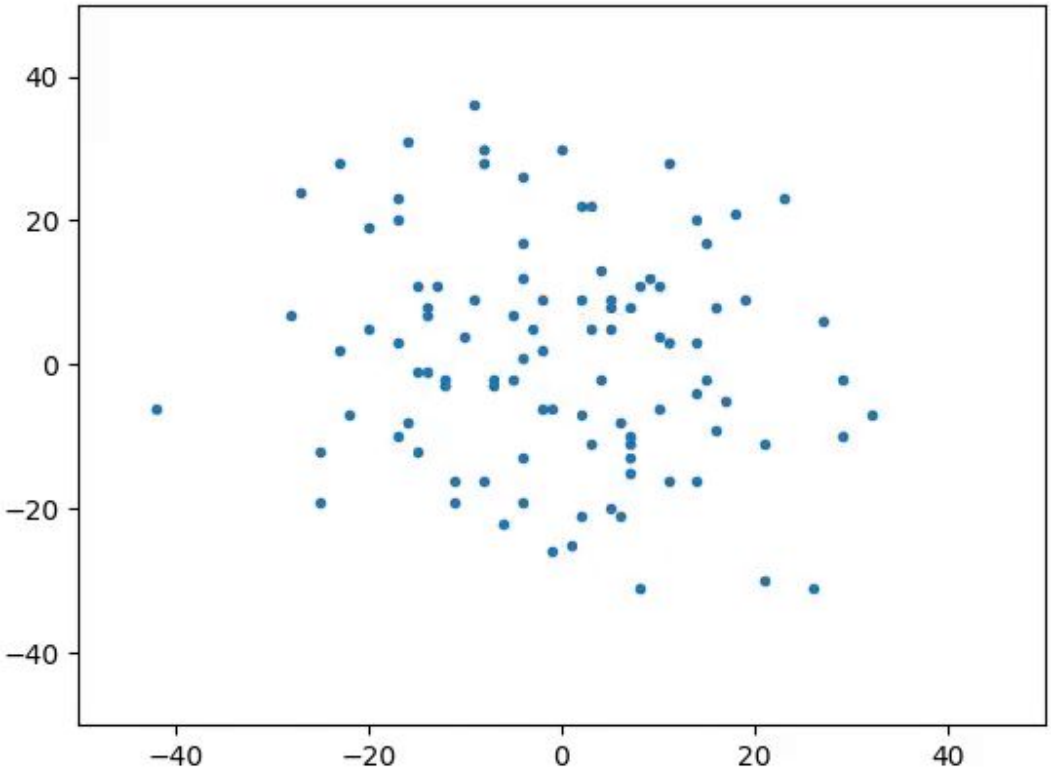
Large Brownian Motion Simulation



Random Walk with N Particles

By: Blake Cole

Brownian Motion or Brownian Movement is a stochastic process resulting in the motion of a particle suspended in a fluid. This project visualizes such motion by simulating a random walk of N particles on a 2 dimensional grid. Classical Brownian Motion experiments involve a particle or particles which are thousands of times larger than the molecules of the suspending fluid. The kinetic energy of the fluid, which corresponds to its internal thermal energy, results in many thousands of elastic collisions with the particle. These collisions result in random impulses causing the particle to accelerate. The fluid's viscous properties cause the particle to decelerate, and the combination of these forces result in the particle's random walk through the fluid. The collisions are inherently random, and the total probability of moving in any direction is zero, since a collision in any direction is equally probable.

The goal of this project is to make a simple model of a random walk involving N particles confined by boundaries. In order to achieve this, a set of N particles is initially created with a pair of x,y coordinates defining each particle. A random step of magnitude 1 is applied in the x and y direction, causing each particle to move in a random walk through the grid. This simulation differs from realistic Brownian Motion since in real life the step size and direction are not discretized and it is invariant in any rotation. This model uses a discretized step size and is only invariant for rotations of 90 degrees. Thus the model is much simpler, however the resulting random walk obeys similar statistics.

There are likely many ways to simulate a random walk on a grid. The program I wrote implements it in the way that makes the most sense to me. The program starts by adding N random numbers to a list. This is done twice, and serves as the initial x,y coordinates for N

particles. These steps could be scaled up or down depending on the size of the grid and number of particles that is desired. Working with N=100 and on a grid that spans -50 to 50 in either direction, I chose to spawn particles within an area between -20 and 20.

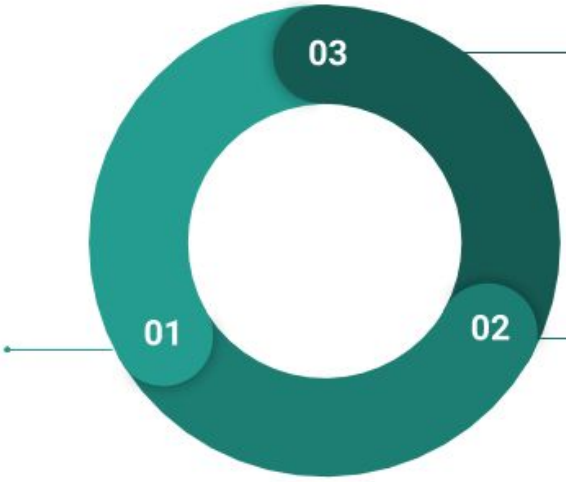
The following steps occur within the main animation function which also serves as a simple loop for animation. Inside a for loop, I define a step for the x direction as a random integer, -1, 0, or 1. In the same way, I define a random step for the y direction. These numbers need to be defined separately otherwise the particle will simply travel in a straight line. The for loop loops over N numbers, selecting each element in the lists of x and y coordinates, and adds the corresponding random step. Instead of appending to the list of numbers, the loop updates each coordinate, such that coordinate=coordinate+random step.

To define the boundary conditions, I added several if statements, such that if the particle's x or y coordinate is equal to the boundary conditions, it subtracts the step that it just made. In effect reflecting the particle.

I set the plotting window to have a constant size, and plot the x and y coordinates for each particle in the window. To animate the plotting I used the matplotlib animate package, along with a function for clearing the previous plot. This way the program redraws the window with the updated particle position each time the animate loop runs, though this can be commented out and has no effect on the program besides covering the screen in particle positions.

Initialize particles

Use a for loop to generate random numbers and append to a list. Creates a random starting coordinates for each particle.



Plot

Plot the particles and repeat the loop

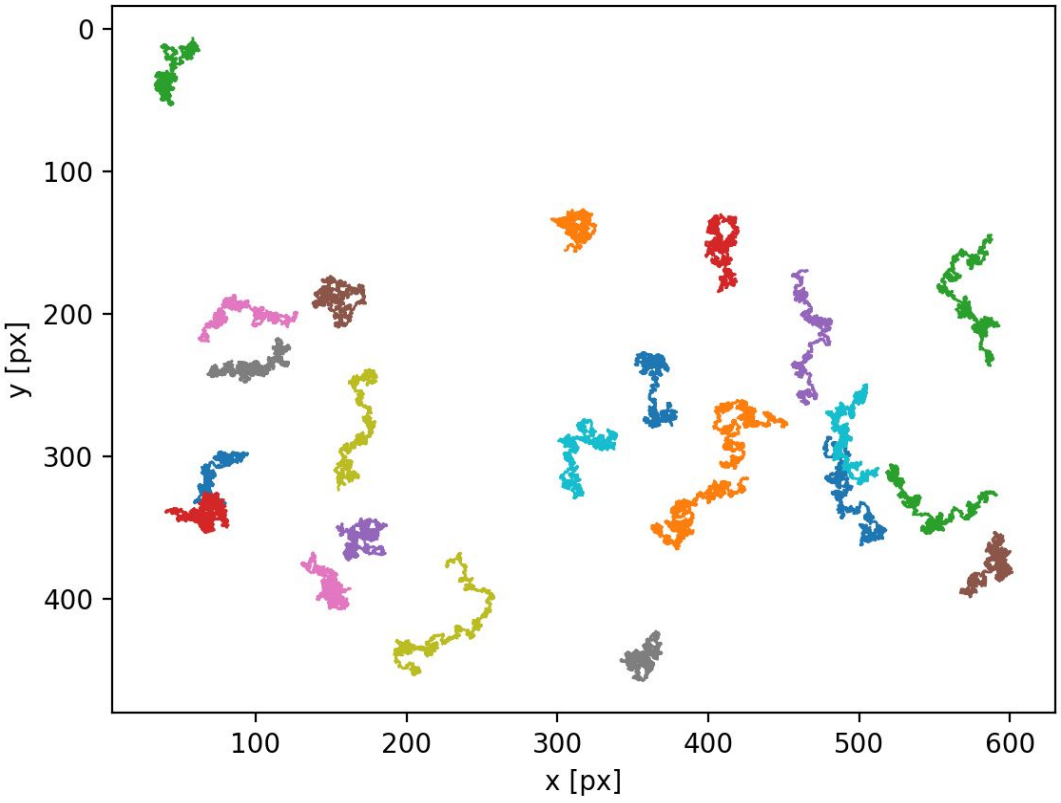
Add Random Step to x and y coordinates

Loop through the list of size N and update add Random Step to coordinate values

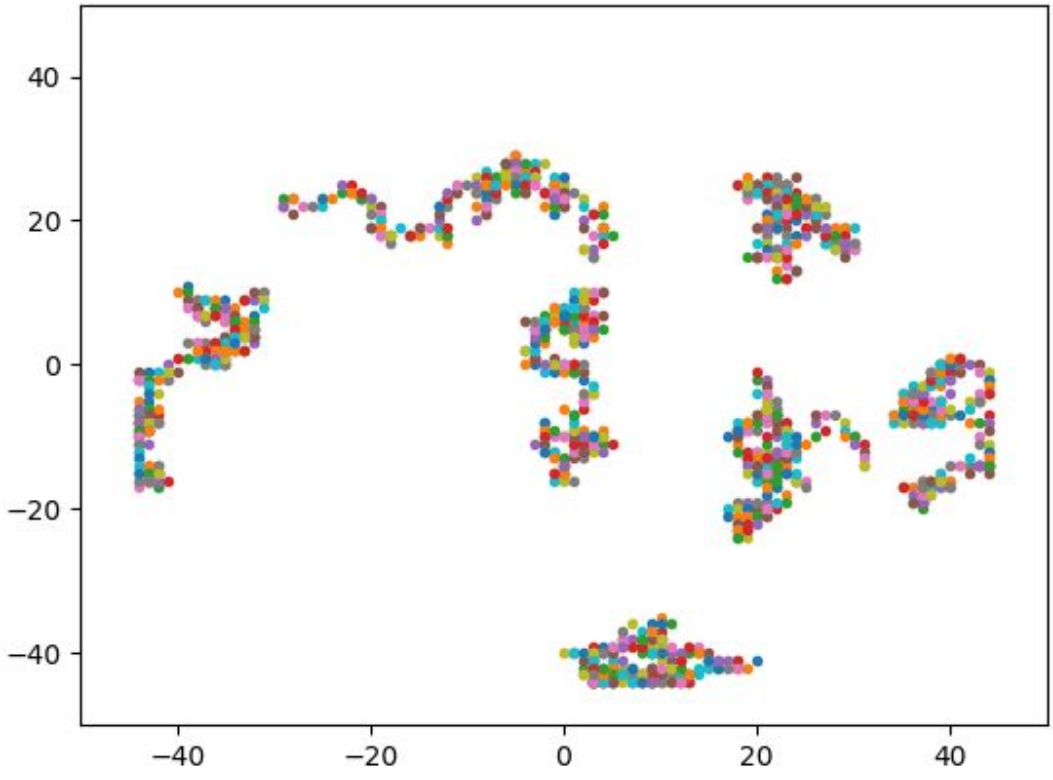
Create Random Step

Use Python random integer to create a random number between -1,1.

My Capstone project involves recording a system of microparticles suspended in fluids of various viscosities then tracking those particle displacements. Below is the displacement graph from particles recorded during an experimental trial in pure water.



The trajectory plots generated by my program look quite similar over a short period of time. Below is one such plot for N=7.



The discretized motion in the simulation seems clear by comparison, as well as the fact that several particles in the simulation are near the hard boundary which does not exist for the experimental data. Comparing the experimental data also shows how true Brownian Motion is continuous and invariant in all directions, resulting in smooth particle displacements. The computer generated data appears blocky by comparison, since it only has 4 directions of movement. Additionally the experimental data was recorded over ~15 seconds, compared to the ~5 seconds for the simulation. This seems to indicate that shrinking the step size as well as adding angular degrees of freedom could improve the realism of the simulation. The experimental data may very well contain an overall drift which is absent in the simulation. Interestingly, since the experiment occurred in a 3 dimensional space there is limited particle to particle collision, which is absent in the simulation as well. It is surprising how well a simple model works, given that the simulated particles do not have a velocity and the grid does not have resistance to motion. I think this reflects the fact that the Brownian Movement occurs from the random collisions which are modelled well as changes to coordinate position.

References:

Catipovic, Marco A., et al. “Improving the Quantification of Brownian Motion.” American Journal of Physics, vol. 81, no. 7, American Association of Physics Teachers (AAPT), July 2013, pp. 485–491. Crossref, doi:10.1119/1.4803529.