Modeling Brownian Motion with Elastic Collisions

Michael-Erik Ronlund
Physics Department, The College of Wooster, Wooster, Ohio 44691, USA
(Dated: December 15, 2011)

In this project, I designed a program to simulate Brownian Motion as the interaction of elastic particles. The program is intended to observe how this simple model would hold up to expected mathematical models by only using calculations for Conservation of Momentum and Energy. While the program has the capability to produce a large system and display effects that appear to be the signature of Brownian Motion, the output for disk position and molecule properties is as of yet nonfunctional, and so I could not acquire a distinct graph of the actual motion to determine its nature.

I. INTRODUCTION

Brownian Motion is an effect named after Robert Brown. Brown noticed that when suspended in water, a grain of pollen would exhibit a strange behavior: it would move and jitter about, seemingly randomly [1]. Brown's original hypothesis about the odd traveling of the pollen was that it was a property of living matter, that living things, even smaller components of them, had some mobility to them. His later studies using non-living particles showed that that was not the case. The movements exhibited by Brown's pollen are actually the result of numerous impacts from the molecules of water surrounding it. As the particle was continually bumped into, it was pushed across the surface.

The motion of the particle is dependent on the molecules of the medium. A simple way of picturing it is that at a very small scale, the motion is the direct result of the sum of each collision with the surrounding molecules. The changes in its motion are based on the exchange of momentum between the particle and molecules. As these collisions occur, the continuous bumps will displace the particle. The direction in which the displacement occurs is random, and based in part on the properties of the medium and the particle [1].

Brownian Motion is an example of a general category of random processes, the study of which can be applied to many disciplines to make predictions. Mathematically, it's motion is a result of the probability of the disk moving in a certain direction over a certain time [7]. One area that relies on it is stock market and finance study. Even though the model isn't a perfect analogy, they both rely on the accumulation of small changes to cause an overall larger result [3]. The model can be applied in other fields as well, such as biology, where it used to understand the movements of microscopic organisms [2].

In order to help understand this process at a basic level, and present a clear visual representation, I aim in this project to create a computer simulation of Brownian Motion, based in part on the model used by Scott Hughes for a similar project [5]. This will mainly rely on techniques and data structures from the standard C++ language library [6]. The program creates an environment of many small molecule-like particles, which interact with a larger disk-like particle. The simulation explores the basic exchange of momentum in determining the motion of the particle.

II. THEORY

The concept of Brownian Motion in this simulation is the seemingly randomized manner in which a disk-like particle will move around in an environment of many smaller particles. To better understand this phenomenon in the simplest physical situation, I created a program that modeled a large disk and many smaller particles, and allow them to interact using conservation of momentum and energy.

To create such a simulation, it would need to be based on an accurate calculation of the resulting velocities of the disk and a particle upon collision. The assumption that all collisions will be elastic makes this easier, as it allows for the combination of the laws of Conservation of Momentum and Conservation of Energy. These are known to be

$$m_1 v_1 + m_2 v_2 = m_1 v_1' + m_2 v_2' \tag{1}$$

for Conservation of Momentum, and

$$\frac{1}{2} m_1 v_1^2 + \frac{1}{2} m_2 v_2^2 = \frac{1}{2} m_1 (v_1')^2 + \frac{1}{2} m_2 (v_2')^2. \tag{2}$$

for Conservation of Energy. It is possible to combine the two equations to find specific equations for the resulting velocities after a one dimensional collision of two objects. These turn out to be

$$v_1' = \frac{v_1(m_1 - m_2) + 2m_2 v_2}{m_1 + m_2} \tag{3}$$

and

$$v_2' = \frac{v_2(m_2 - m_1) + 2m_1 v_1}{m_1 + m_2}. \tag{4}$$

Now, this gives results for when two objects collide one dimensionally, but in a simulation of Brownian motion, the collisions can have two dimensional components. In order to use these equations for the calculations, it is necessary to find a way to view any possible collision between the disk and a particle from a frame of reference such that the collision is one dimensional.

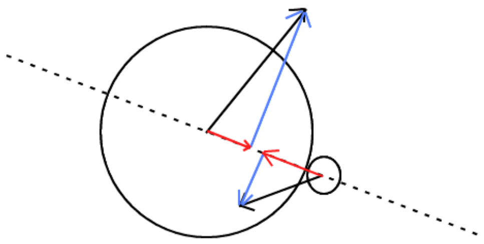


FIG. 1: Diagram of the vectors used in calculation of velocity for use in Conservation of Momentum.

One way to do this is to consider, when a collision occurs, the line connecting the centers of the particle and disk. The respective velocities of the disk and particle can be viewed as being made of two components: one along the connecting line, and one perpendicular to it, as shown in Fig. 1. The particles can be thought of as reflecting off each other along an axis orthogonal to the connecting line. As such, the velocity components along that axis, denoted $v_{\parallel}$ as they are parallel to the reflecting line, will remain unchanged, and all that will be affected are the components along the connecting line, denoted $v_{\perp}$ as they are perpendicular to the reflecting line.

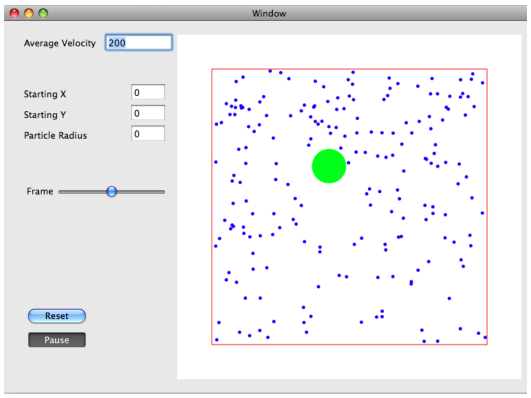
Now Eq. 3 and 4 can be used with the $v_{\perp}$ components as the starting velocities. This will give a new perpendicular component of velocity to both the disk and particle, which when added to their respective unchanged $v_{\parallel}$ components, gives the new total velocity vectors of the disk and particle.

Brownian Motion has a distinctive quality that, for a number of N collisions, the root mean square of the distance traveled by the large particle will be approximately $\sqrt{N}$ [1]. This can be checked by observing the disk's travel through the medium, and calculating its distance every few steps and comparing it to a recorded number of collisions.

III. PROGRAM ASSEMBLY

To simulate the phenomenon of Brownian Motion, I created a program that set up a large particle and a set of 200 smaller particles. The basis of the program's functionality revolved around the interactions of the particles. In my simulation, the interactions were based on the assumption that all collisions were elastic, and therefore conserved both momentum and kinetic energy, as well as the assumption that the smaller particles were point masses and did not interact with each other. There are three main portions to the program that operate behind the scenes of the simulation: the initialization component, the evolution component, and the collision component.

The initialization part of the program determines the starting conditions of the simulation universe. This involves setting up the boundary conditions of the universe, the starting mass, radius, position and velocity of the large particle, and the mass and starting positions and velocities of the small particles. The boundaries of the universe are defined simply by user input. The large disk is by default set to position coordinates (0,0) with a velocity of 0 $\frac{m}{s}$. The many smaller particles are given their starting conditions, position and velocity, by a pseudorandom number generation from the C++ arc4random() function. Before the particles themselves are created, two arrays of vectors are set up, one array for the position vectors and one for the velocities. These arrays contain n elements, where n is the number of particles that will be placed in the universe, so that each entry corresponds to a certain particle. For the position array, the coordinates are generated as random numbers, limited by the boundary size of the universe so that the particles are not created outside. The velocities are defined by assigning a random x component, which can have a magnitude from 0 to the default maximum velocity of 200 $\frac{m}{s}$. This velocity did not have particular mathematical significance, but was a compromise of the largest amount of particles before I began noticing a collision error, which will be discussed in the Future Work section. These are also randomly assigned to be either negative or positive. The y component of the velocity is then determined by the Pythagorean Theorem such that the magnitude of the entire velocity vector equals 200 $\frac{m}{s}$, and is again randomly set to be negative or positive. In this way, each small particle has the same magnitude of velocity but different position and velocity directions. Once these arrays are created, the particles are then created and assigned their values from the arrays, and placed in the universe.



Average Velocity 200
Starting X 0
Starting Y 0
Particle Radius 0

FIG. 2: Photo of the final simulation during motion.

Once all the starting conditions are set up, the program then begins running the simulation. This animation is controlled by an evolution function, which determines how the particles will move and behave as time advances. This is broken into three steps. First, both the particles and disk are checked to see if they are going to collide with a wall. In the event of a wall collision, the component of velocity perpendicular to the wall is reversed, while the parallel component is left alone. Secondly, the program checks if any of the small particles are about to collide with the large disk. This is done by determining if their move forward will result in the distance between the particle and disk's centers being be less the sum of the radii, which would indicate that the particle would move inside the disk. If both of these tests pass, the particles and disk are moved forward based on their current velocities. If the particle or disk hits a wall, it's component of velocity perpendicular to the wall is reversed, and the parallel component is left alone. If a collision between a particle and disk occurs, the program moves into the colliding step.

When the colliding step has been activated, the program proceeds to make a calculation to find the new velocities of the disk and particle. This is done by finding the components of velocity of the disk and particle along the line joining their centers, and using them in a one dimensional conservation of momentum calculation, as discussed in the Theory section above. Once the new velocities are found, the program has finished its calculation and returns to the evolution step to continue advancing the simulation.

The program displays the movement of the particles so the user can see how the collisions affect the disk and it's momentum, as illustrated in Fig. III.

IV. CONTINUATION AND FUTURE WORK

While the program generates a simulation that seems to be accurate in terms of the motion of the disk, there are some abnormalities. The most noticeable is the fact that every so often, the disk will appear to "eat" a smaller particle. That is to say, the the particle will pass inside the disk rather than colliding as it should. This happens only rarely, but often enough to merit investigation into how this effects the overall accuracy of the simulation. One hypothesis for the origin of this event is that in the evolution function, there is an incorrect way of updating the positions of the particles. While it is difficult to tell with so many particles bouncing around, it is possible that this occurs when the disk is struck by more than one smaller particle at once. The program does not have a specific instruction as to how to deal with that situation, and so it may be responding inappropriately. Another possibility is that it originates from the order in which the tests are conducted.

As for continuation of the project and future directions that it could go, there are several options. I would like to collect more data from the the current simulation, hopefully after repairing the inconsistencies mentioned above. I would like to check how the system behaves when the number of particles, their average velocity, and universe size are variable.

Looking to where the project can go once the basics are taken care of, one idea is the situation where the large disk has a simulated high temperature, greater than the average kinetic energy of the smaller particles [4]. The simulation could then attempt to determine how, if at all, the outcome would change as the disk gradually transferred energy to the particles. In this case, it might be valuable to allow the small particles to interact with each other as well as the disk. One concern about doing this is that, under the current way in which the program checks collisions, allowing the particles to bounce off each other would change the number of checks needed from n to $n!$, as each particle would have to check all of the others to see if it would bounce. This would greatly increase the runtime of the program, meaning a drastically slower speed of simulation.

A possibility for dealing with that increase is to use a different algorithm to determine the collisions. One method would be to analyze the particles as they advanced through space time, and find where they would intersect walls or each other, then recalculate after each bounce. By doing the this kind of simple geometric calculation, the runtime of the program would return to the speed it is at now, or better, allowing for a very fast calculation of a much more complicated system.

V. ACKNOWLEDGEMENTS

I would like to thank Dr. John Lindner, for his encouragement and help with finding the appropriate algorithms for the collision calculations.

[1] D. S. Lemons. An Introduction to Stochastic Processes in Physics. (The Johns Hopkins University Press, Baltimore, 2002).

[2] G. Li. Amplified effect of Brownian motion in bacterial near-surface swimming. *Proc. Natl. Acad. Sci. USA* **105** 18355-18359 (2008).

[3] M. J. Lax. Random Processes in Physics and Finance. (Oxford University Press, Oxford, 2006).

[4] R. M. Mazo. On the theory of Brownian motion. VII. A hot particle in a dense medium. *Journal of Chemical Physics* **60** 2634-2637 (1974).

[5] S. Hughes, J. F. Lindner, W. Ditto. Chaotic Brownian Billiards, Centennial Meeting of the American Physical Society (Atlanta Georgia, March 1999).

[6] T. Budd. Data Structures in C++ Using the Standard Template Library. (Addison Wesley, Reading, MA, 1998).

[7] V. G. Kulkarni. Introduction to Modeling and Analysis of Stochastic Systems. (Springer New York, New York, 2011).