Simulating Three-Dimensional Walks in Closed Environments

Theo Gerst, Dhruv Goyal, Lucas Zhou, Humza Mahmood

Our Idea

Life has selected cell shape and size for billions of years, based on the efficiency of diffusion and other critical processes. We were partly inspired by the fact that small cells are better for supporting life rather than large cells, as the surface area to volume ratio is higher when the cells are smaller. This higher ratio allows for more entry sites for materials necessary to support the cell, and the small volume means that transportation of these materials across the cell is quicker and requires less energy. Here, we examine random walks of particles (such as proteins) in three dimensions from one end of a "cell" to the other.

Question: Does cell shape have a significant impact on the time it takes for particles to move across the cell if the particles move with completely random motion?

We'll be running four test cases:

- Case 1: Sphere with radius 12.41, particle moving from <-10, 0, 0> → <10, 0, 0>
    - Volume: 8006 units^3, distance: 20
    - Control case
- Case 2: Sphere with radius 12.41, particle moving from <-12.41, 0, 0> → <12.41, 0, 0>
    - Volume: 8006 units^3, distance: 24.82
    - Experimental case, variable being tested: distance
- Case 3: Sphere with radius 10, particle moving from <-10, 0, 0> → <10, 0, 0>
    - Volume: 4189 units^3, distance: 20
    - Experimental case, variable being tested: volume
- Case 4: 20 * 20 * 20 cube, with particle moving from <-10, 0, 0> → <10, 0, 0>
    - Volume: 8000 units^3, distance: 20
    - Experimental case, variable being tested: shape

There are 3 variables we are altering in these experiments: cell shape, cell volume, and distance that the "particle" needs to travel. Our control has the shape of a sphere, volume ~8000, and distance 20 **units (unspecified)**.

We hypothesize that the sphere with radius 10 (Case 3) will allow the particle to reach the target in the shortest number of steps (time), as the volume is almost two times smaller than the other cases. While the sphere with radius 12.41 (Case 1) is the same shape, its increased size should make the protein take much longer to reach its target. We predict that the sphere with radius 12.41 (Case 1) and the 20*20*20 cube (Case 4) will have no significant difference in the number of steps needed to reach the target, as their volumes are the same, and the distance the particles need to travel are the same. Finally, because the particle in the sphere in Case 2 needs to travel a slightly increased distance, we predict it should take the most steps for the particle to travel from the starting point to the ending point.

Code

First, we import the necessary libraries: random (for generating unit vectors with random angles), numpy (for trig functions), and matplotlib (for graphing our histograms).

```
import random as r
import numpy as n
import matplotlib.pyplot as p
```

Next, we'll define a method for generating a random unit vector in three dimensions, uniformly distributed for theta, phi, x, y, and z. We ran tests prior to this experiment to confirm that the distribution was indeed random. The rndwalk() method, defined afterwards, runs a random walk from a **start** position until it gets within a certain **radius** of a given **target**. The method also includes a **rule** parameter, which defines what the particle can and can't do (for example, a spherical cell disallows sqrt(x^2 + y^2 + z^2) from being larger than the radius).

```
from mpl_toolkits import mplot3d
def vec():
    phi = r.uniform(0, n.pi*2)
    costheta = r.uniform(-1, 1)
    theta = n.arccos(costheta)
    x = n.sin(theta) * n.cos(phi)
    y = n.sin(theta) * n.sin(phi)
    z = n.cos(theta)
    return (x, y, z)
def rndwalk(start, target, rule, radius):
    pos = [start[0], start[1], start[2]]
    numsteps = 0
    while n.sqrt((pos[0] - target[0]) ** 2 + (pos[1] - target[1]) ** 2 + (pos[1] - target[1]) ** 2) > radius:
        res = vec()
        while not rule(pos[0] + res[0], pos[1] + res[1], pos[2] + res[2]):
            res = vec()
        numsteps += 1
        pos[0] += res[0]
        pos[1] += res[1]
        pos[2] += res[2]
    return numsteps
```

This next method runs a test for a certain random walk and plots a histogram of the results. The function takes the same parameters as rndwalk(), along with **numwalks** (the number of walks to be added to the data) and **shapeName** (the name of the enclosure, to be used for formatting purposes). This code also runs a test for the 20 * 20 * 20 cube.

```
def runtest(start, target, rule, radius, numwalks, shapeName):
    dists = []
    for i in range(numwalks):
        dists.append(rndwalk(start, target, rule, radius))
    p.hist(dists, 20)
    p.title(f"Walk distance distributions for {shapeName} over {numwalks} walks")
    p.xlabel("Walk distance (steps)")
    p.ylabel(f"Probability (out of {numwalks})")
# 20 * 20 * 20 cube
runtest((-10, 0, 0), (10, 0, 0), lambda x,y,z: n.abs(x) <= 10 and n.abs(y) <= 10 and n.abs(z) <= 10, 1, 10000, "cub
```

This next cell runs a similar test for a sphere with radius 10…

```
# r = 20 sphere
runtest((-10, 0, 0), (10, 0, 0), lambda x,y,z: n.sqrt(x ** 2 + y ** 2 + z ** 2) <= 10, 1, 10000, "sphere(r = 20)")
```

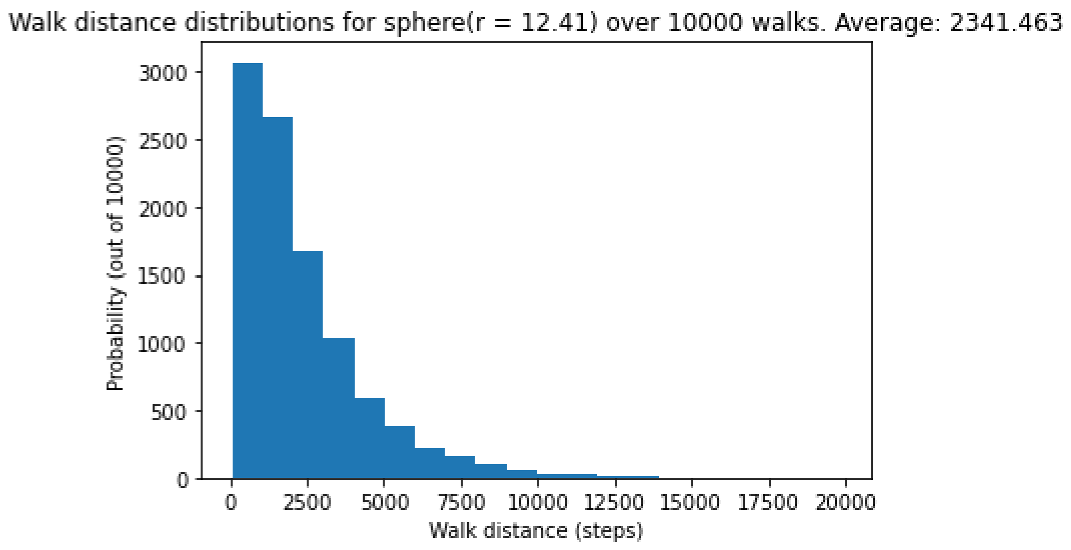
... this one runs for a sphere with radius 12.41 and distance 20 …

```
# equal volume sphere from -10 to 10
runtest((-10, 0, 0), (10, 0, 0), lambda x,y,z: n.sqrt(x ** 2 + y ** 2 + z ** 2) <= 12.41, 1, 10000, "sphere(r = 12.
```

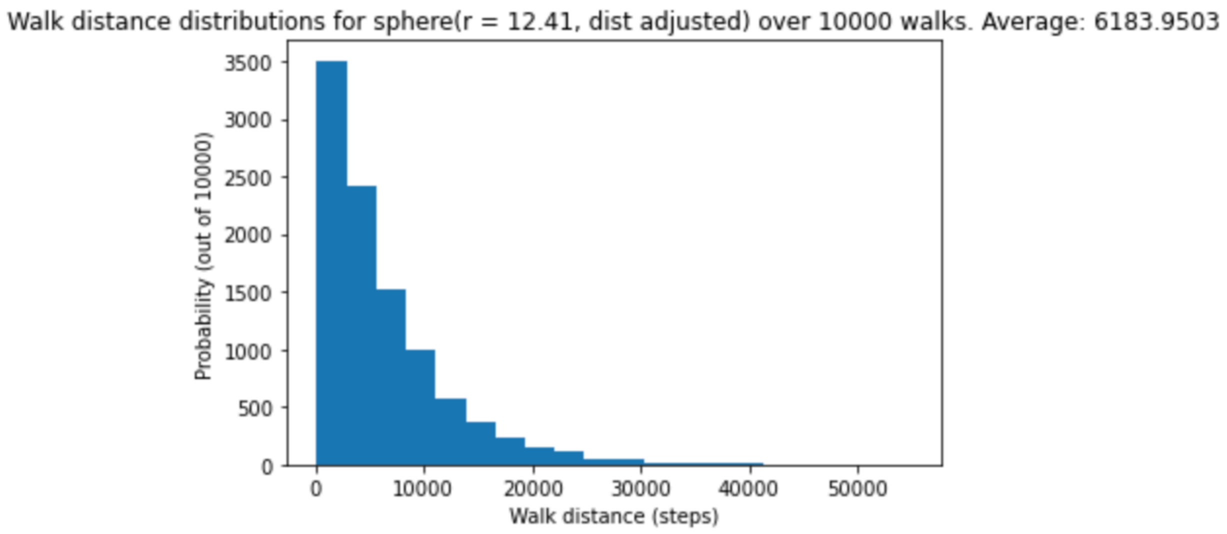
... and this last one runs for a sphere with radius 12.41 and distance 24.82. All of our tests run for 10,000 walks, which allows for accurate data and takes roughly 7 minutes to run per test.

```
# equal volume sphere from -10 to 10
runtest((-12.41, 0, 0), (12.41, 0, 0), lambda x,y,z: n.sqrt(x ** 2 + y ** 2 + z ** 2) <= 12.41, 1, 10000, "sphere(r
```

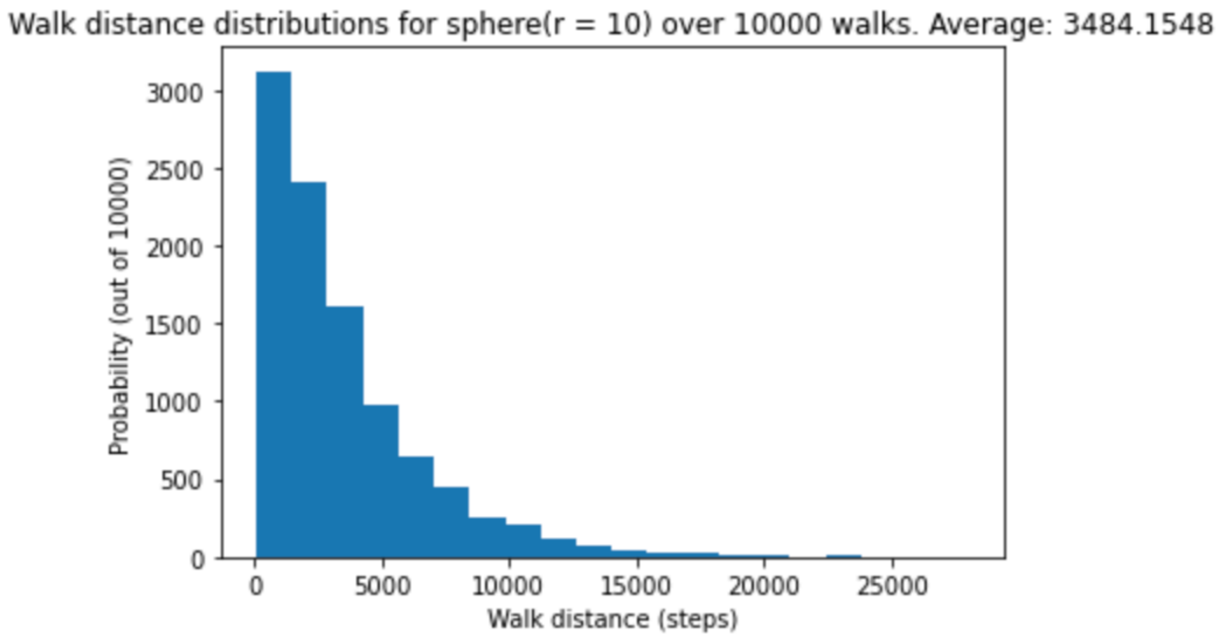
Altogether, each test generates a separate histogram, rendered with 20 containers. Here are our results:



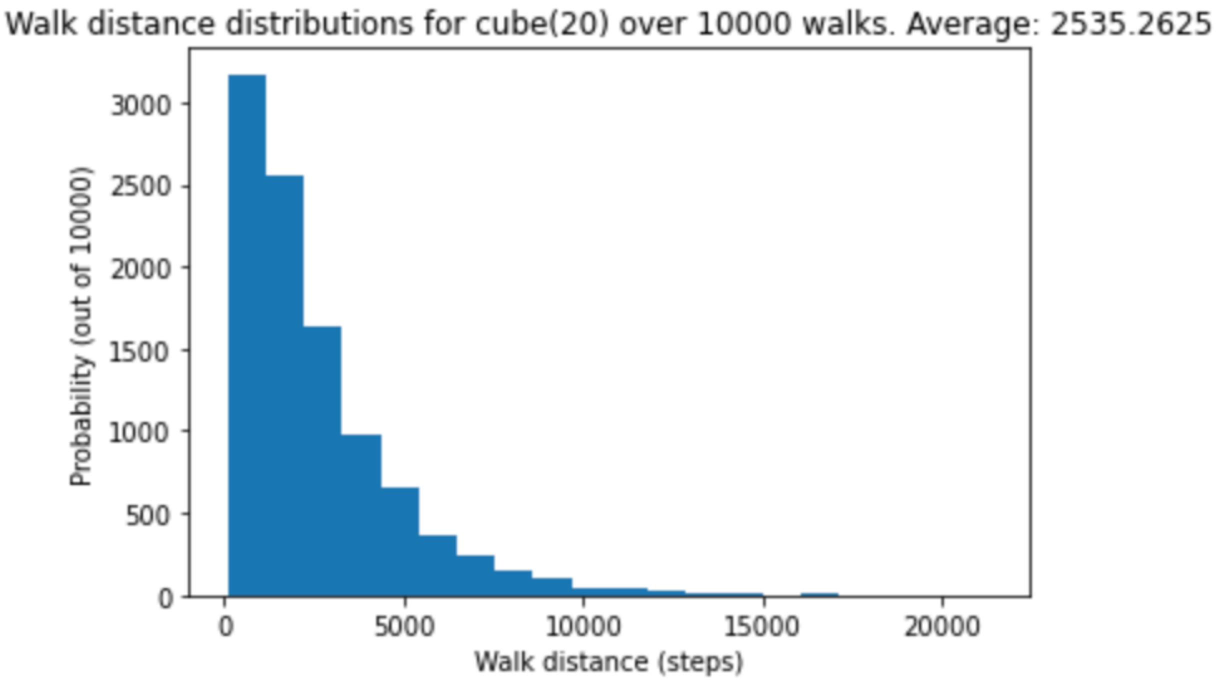
The control case sphere (Case 1, control case) had data primarily distributed from 0 to ~1000 steps with a right skew.



The large distance sphere (experimental case with distance adjusted) was primarily distributed from 0 to ~5000) steps with another right skew. The average step count was significantly higher than that of the small sphere and the cube, which makes sense because the distance is greater.



The small sphere (experimental case with volume adjusted) was primarily distributed from 0 to ~3000 steps with a significant right skew. The average step count was ~3000, which is a surprisingly large amount for a small sphere with a radius of 10 and half the volume of the 20-20-20 cube.

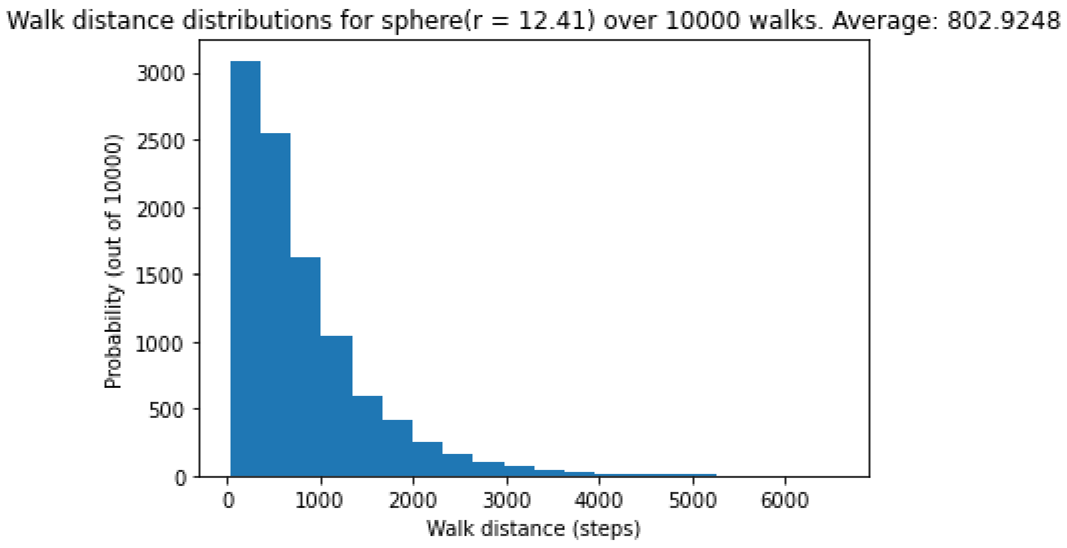


The cube (experimental case with shape adjusted) was primarily distributed from 0 to ~2500 steps, with a significant right skew. The average step count was ~2535, much less than that of even the smallest sphere.

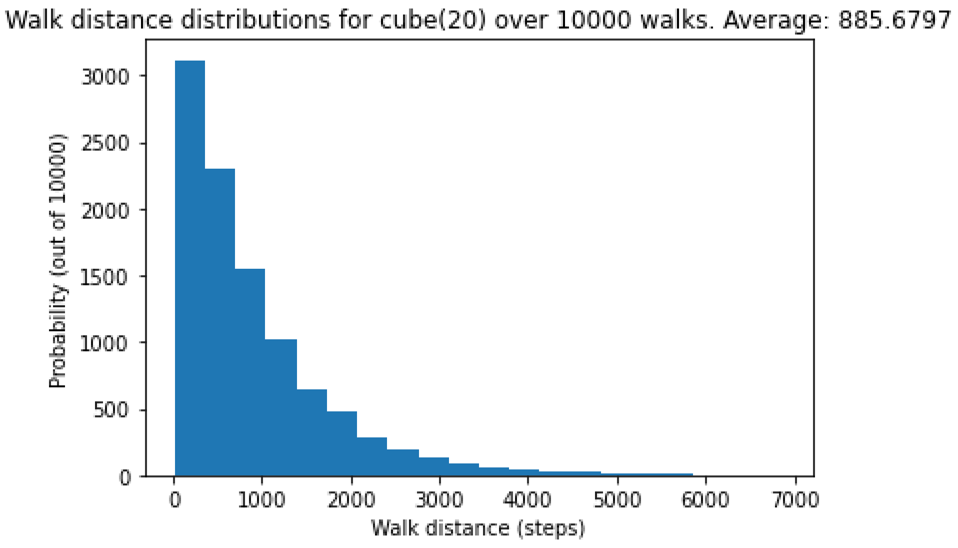
Errors

1. The program is designed to stop when the walker reaches within a distance of 1 unit of the final target. This essentially means that the walker needs to enter an imaginary sphere of radius 1 around the ending point. However, because our final target is on the edge of our object (face of a cube, edge of sphere) the size of the portion of the sphere that stops the walker is slightly smaller in the sphere tests because the portion cut off is slightly more than the 50% removed in the cube tests. This leads to a slightly lower probability of the walker reaching its destination at any given walk, which slightly increases the number of steps needed.
2. In Case 1, the sphere around the final point is completely inside the larger sphere which the walker is traveling through. This means that the area in which the walker can enter to finish the program is twice the size of the corresponding cube test, thus lowering the number of steps needed for it to find the target.

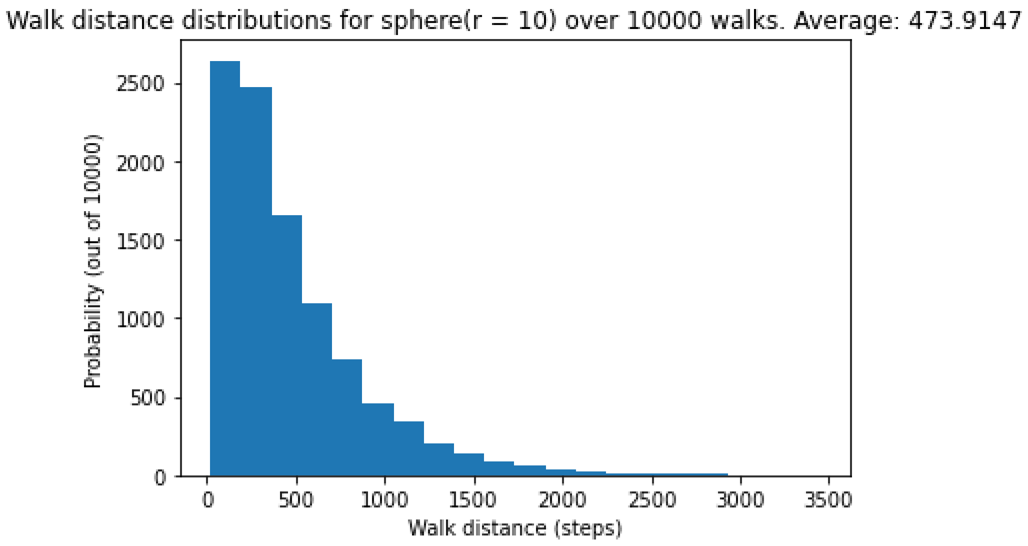
Correcting for Errors



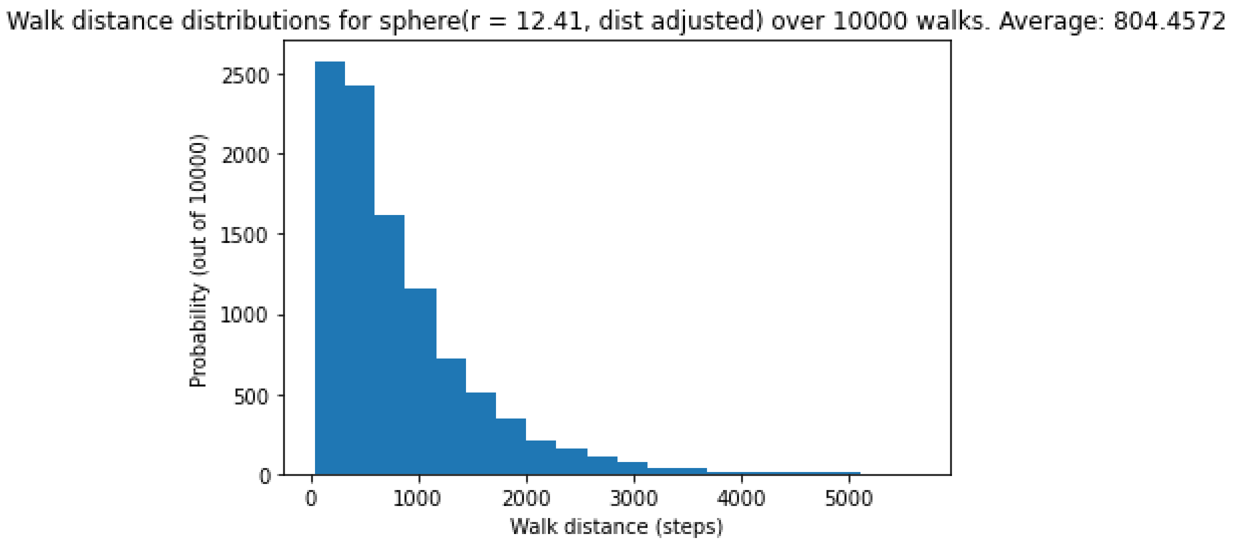
The regular-sized sphere (Case 1) was primarily distributed from 0 to ~400 steps with a significant right skew. The average step count was ~803.



This distribution shows the walk distance for particles in a **cube** moving from <-10, 0, 0> to the center (<0, 0, 0>), which should negate our error with the target sphere being cut off by the outer container. The average distance was much lower than previously measured, because the distance required was 10 rather than 20.



The small sphere (experimental case with volume adjusted) was primarily distributed from 0 to ~400 steps with a significant right skew. The average step count was ~473, which was the expected result for a sphere with a small radius of 10 if the random walks moved from <-10,0,0> to the center, <0,0,0>, so the particle would move much faster as the radius decreases.



Our final test (the large sphere with extra distance) ran in an average of 804 steps per walk. This was not significantly different from our other large sphere test, and so we can conclude that distance adjustments actually don't affect the average walk length, as long as they are relatively small.

Conclusions:

Based on our second round of results (after adjusting for error), volume has a significant impact on the distance traveled during random walks. The larger (control) sphere took roughly 1.7x as long to transport a particle from one end to the center compared to the smaller sphere, which had around half of the volume. In terms of the shape of the cell, the cube had a small but perceivable effect on distance distribution. It took roughly 803 steps for control walks to reach the center, while it took 886 steps for cube walks to reach the center. As for distance, it seemed to have almost no effect on random walk distance (803 versus 804 steps on average). We imagine that this effect would've been more visible had the adjusted distance been greater.