WaffleWalk: Exploring 2D Random Walks in Restricted Space

Eason Wang[1]*
[1]*Department of Physics, University of Arizona, Tucson, AZ 85721 USA*

17 June 2024

ABSTRACT
In this work, I explore how random walks behave in a space where certain territories are prohibited. Specifically, I explore how random walk particles spread inside a waffle grid (WaffleWalk), where there are evenly spaced square regions where the particle is not allowed to reach. I compare certain properties of this WaffleWalk with that of an unrestricted random walk, such as the time dependence of displacement and the shape of the random walk distribution. I also discuss potential uses of exploring this type of problem.

1 UNRESTRICTED RANDOM WALK

First I simulate a normal 2D unrestricted random walk. I start with 10,000 particles at the origin. For each timestep I evolve every particle by moving it 1 unit of distance at a random angle. Figure 1 shows the progression of the simulation at selected timesteps. The particles spread out as one might expect for a random walk.

1.1 Time Dependence

To verify that the time dependence of the average displacement $\langle r\rangle$ is $\sqrt{t}$, I plotted $\langle r\rangle$ as a function of time (top of Figure 2) and fitted a power law curve $\langle r\rangle = t^a$ to it, where I got $a = 0.4797$, rather than the 0.5 expected for a square root relation. This is likely due to my random walk algorithm being only an approximation to diffusion; in reality particles would have different velocities, so they would move different distances at every timestep.

This does not necessarily tell us that our random walk model is inaccurate, though. When I fitted for a using different segments of my $\langle r\rangle$ *vs.* t data, I got a values that approach 0.5 as timestep goes up, as shown in the convergence plot in the bottom plot of Figure 2, meaning that our model approaches a $\langle r\rangle \propto \sqrt{t}$ relationship as the simulation runs for longer times. Furthermore, even the a values at lower timesteps are sufficiently close to 0.5 for the purposes of this work.

1.2 Shape Characterization

For purposes that will become more significant in the next section, I characterized the shape of the particle distribution at any given timestep. The goal is to determine how circular the distribution is. I selected all particles outside the average displacement $\langle r\rangle$, and conducted the following routine:

- Bin selected particles into 10 equal angular bins, as shown in top of Figure 3.
- For each bin i, calculate the average displacement $\bar{r_i}$.
- Calculate the difference in average displacement between each adjacent bin r_i and r_{i+1}:

$$\Delta r_i = |r_i - r_{i+1}| \tag{1}$$

- The circularity C is defined as

$$C = \frac{\langle \Delta r\rangle}{\langle r\rangle}, \tag{2}$$

where $\langle \Delta r\rangle$ is the average Δr_i across all bins.

The progression of circularity for the unrestricted random walk is shown on the bottom of Figure 3. As expected, circularity stays low and roughly constant, telling us that the shape of the distribution stays roughly circular. The waffle grid random walk will show more interesting results for circularity.

2 WAFFLE GRID RANDOM WALK

For this section, I simulated the same random walk, but restrict the particles from going into areas that form a waffle grid pattern. That is, I set squares of certain sizes separated by an equal amount in 2D space, and coded my particles to bounce off the edges of these squares. How I coded the collisions is simple: if the particle ends up going through a wall after its usual displacement, I reflect its motion with the wall as the axis. For example, if a particle goes through a vertical wall, I reflect its x displacement by its horizontal distance to the wall. There are two free parameters for the waffle grid pattern that I am able to adjust: the size of the grids and the spacing between the grids. Figure 4 shows the progression of the particle density distribution at different timesteps.

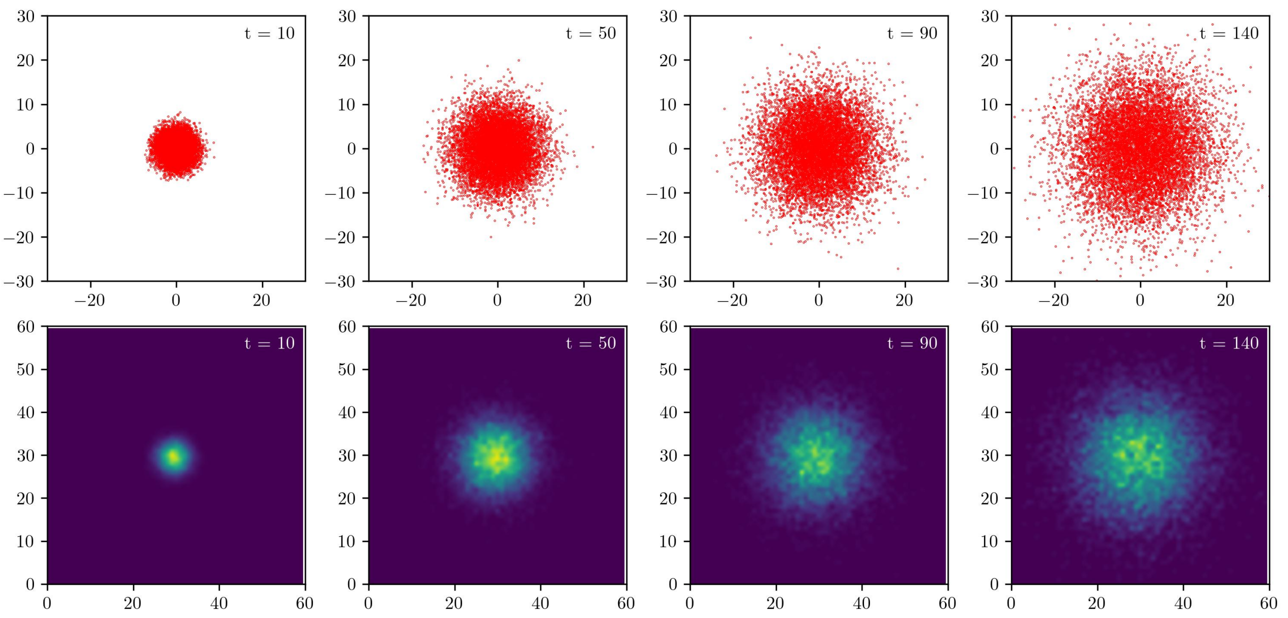

Figure 1. Plots showing the progression of the unrestricted 2D random walk, at increasing timesteps from left to right. Top row shows the distribution of individual particles, and bottom row shows the density distribution of these particles.

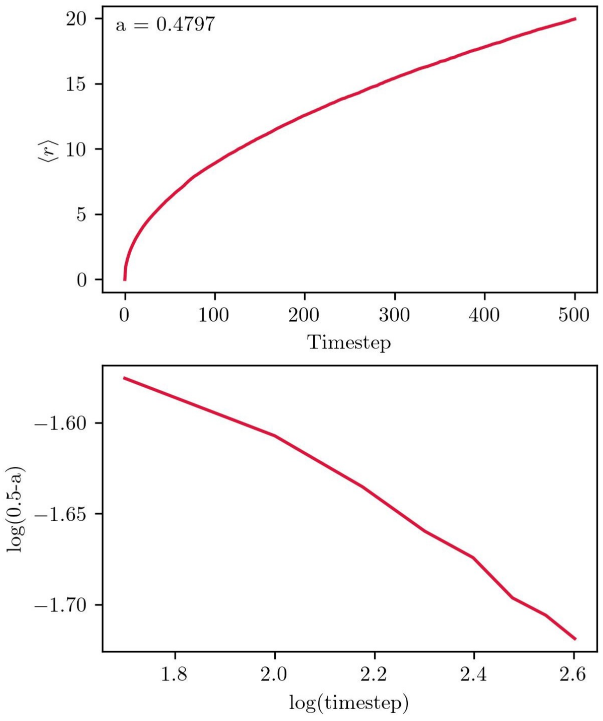

Figure 2. Top: average displacement of particles as a function of time. The value of a on the top left is the power relationship of the plot. Bottom: power relationship between average displacement and time as a function of time. y-axis is the difference between 0.5, which is a perfect $\sqrt{t}$ dependence, and the fitted power law. We see a steady decrease in the log-log plot, indicating that the particle simulation approaches a $\sqrt{t}$ dependence the longer it runs.

2.1 Time Dependence

The time dependence of random walks within a restricted space will be interesting to explore, as this result will be very hard, if not impossible, to derive analytically. For

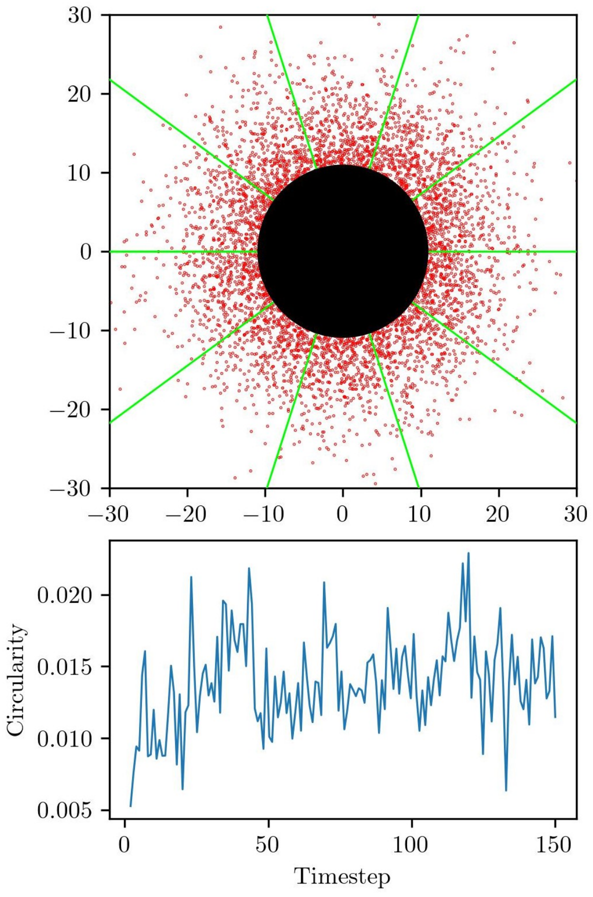

Figure 3. Top: how particles are split into bins for the purpose of calculating circularity. Particles blocked out by the black circle are within $\langle r \rangle$ at that timestep. Green lines show where the bins are split. Bottom: plot of circularity of the distribution as a function of time.

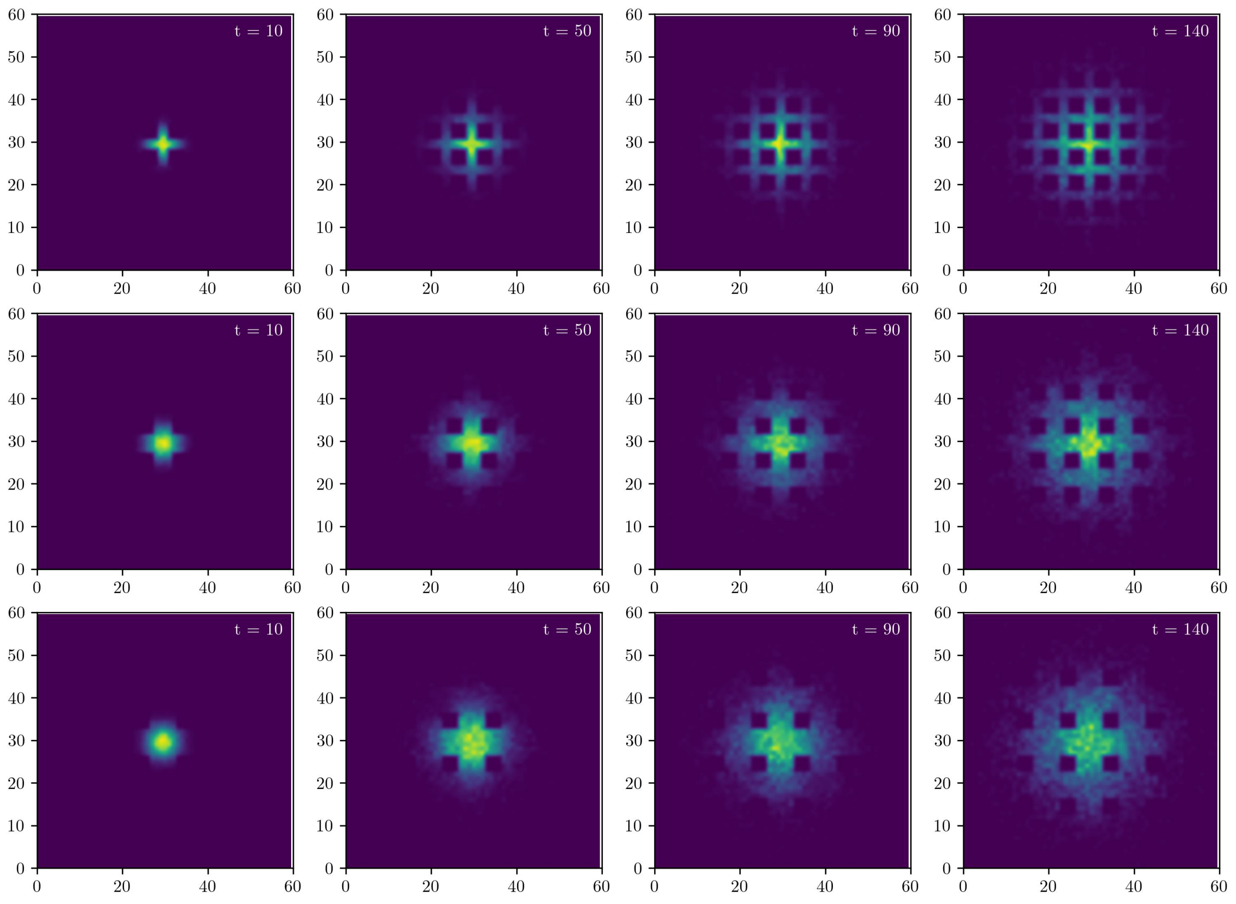

Figure 4. Plot showing the progression of WaffleWalks with different grid parameters. Top to bottom row shows grid spacings of 2, 4, and 6 in order, while grid sizes are kept at 4.

both parameters of my waffle grid, grid size and grid spacing, I fitted a power law for parameter values of 2, 4, and 6, while keeping the other parameter at 4. The power of time dependence a as a function of parameter values is plotted in Figure 5. We see that for both cases the time dependence is below 0.5, and also below that of our unrestricted random walk. This could make sense as there are walls obstructing the movement of particles, making them on average spread slower. We can also see that a increases as grid spacing increases, while it decreases as grid size increases. This also makes sense when we consider the limits. As grid spacing goes to infinity, the particles approach unrestricted movement, so we should expect a to approach 0.5, and as grid spacing goes to 0, a should also approach 0, as the particles would become trapped at the origin. The opposite case is true for grid sizes.

This exploration, when done on a larger scale, could potentially be useful for analyzing diffusion in complex spaces, such as how heat diffuses through a thermally conducting medium intercepted by regions of insulation. If the space arrangement is patterned as it is in a waffle grid, one could possibly use the waffle grid model as an approximation for the real case. One could also potentially study the optimal arrangement of insulators within a thermally conducting medium to efficiently and sufficiently slow down heat diffusion.

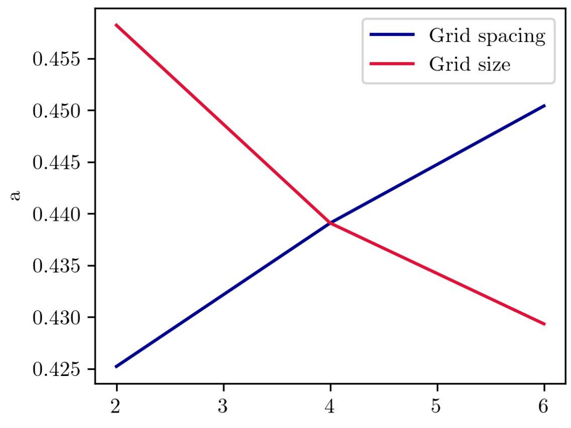

Figure 5. How the power of time dependence a changes with waffle grid parameters. Blue plot shows how a changes with grid spacing, and Red plot shows how a changes with grid size.

2.2 Circularity

I will now explore how the shape of the WaffleWalk distribution differs from that of the unrestricted random walk. Using the same definitions of circularity as in Section 1.2, I calculated the circularity of WaffleWalks of different grid parameters at each timestep, and then plotted the circularity as a function of the average displacement of particles, as shown in Figure 6.

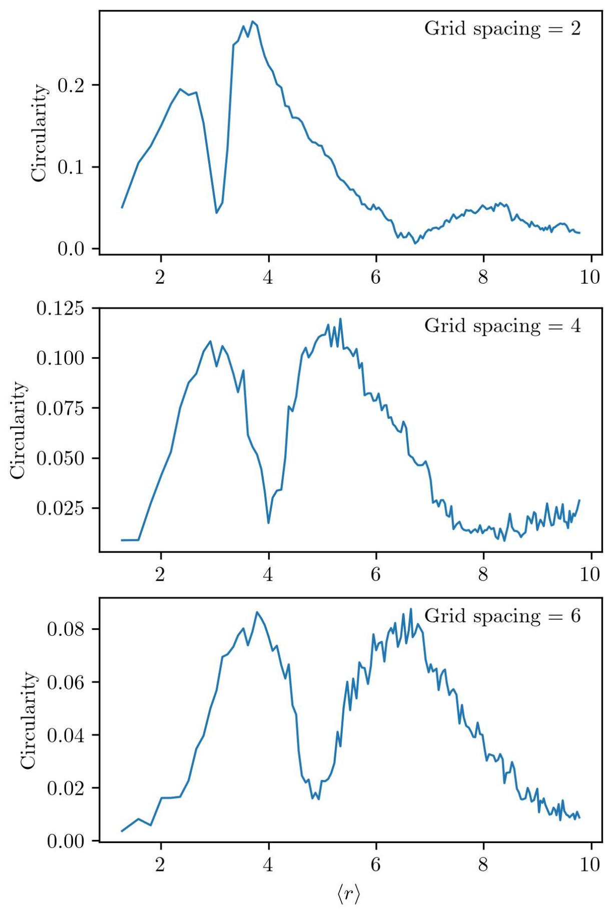

Figure 6. How circularity of the particle distribution changes as a function of average displacement $\langle r \rangle$, for different grid spacing values (labeled on top right of each plot).

One feature we see in these plots is the instability in the beginning. Comparing to the images in Figure 4, the peaks of circularity seem to coincide with when the distribution is spreading out within a "cross", while the troughs coincide with when the distribution is spreading within the "square" surrounding that cross. This shows that the measure of circularity I came up with is quite accurate, as most people would associate a square as being closer to a circle than a cross shape. We also see the peaks decrease in amplitude and appear at larger radii as our grid spacing goes up, which makes sense as the particles would have more unrestricted space to diffuse through before hitting their first wall.

Another feature of note is the circularity approaching 0 at high $\langle r \rangle$. This is to be expected, as the grids will appear finer as we zoom out, approaching what would seem like unrestricted random walks again, except with weaker time dependence, as I've shown in Section 2.1. This result could be useful if we ever want to design a medium that spreads heat evenly on a large scale but slower than typical diffusion.

3 CONCLUSION

This work provided a basic exploration of how random walks navigate a restricted space, specifically the waffle grid space. The main takeaways are 1. time dependence gets weaker as obstacles become more prominent, and 2. distribution shape approaches circular as it spreads out, no matter the grid pattern.

There are several paths that an expansion of this project could take. One could explore different grid patterns and grid shapes, such as shifting every row of grids by half a grid size, or using triangular or circular obstacles. One could also develop this into something approximating a fluid simulation, such as applying a "hydrogen-bond" force between particles to attempt to more realistically simulate the movement of waffle batter on a griddle (this was attempted in this project but with limited success).

APPENDIX A: BASIC CODE TO GENERATE RANDOM WALK

```
import numpy as np
# Initialize particles
N = 10000
T = 500
X = np.zeros((N, T))
Y = np.zeros((N, T))

# Evolution array
r = 1
Theta = 2*np.pi*np.random.rand(T-1, N)
dX = r*np.cos(Theta)
dY = r*np.sin(Theta)

# Evolve particles
for i, x in enumerate(dX):
    X[:, i+1] = X[:, i] + x
for j, y in enumerate(dY):
    Y[:, j+1] = Y[:, j] + y
```

APPENDIX B: COLLISION CHECKER FOR WAFFLEWALK

```
left_bool_x = np.zeros((N, N_grid))
left_bool_y = np.zeros((N, N_grid))
right_bool_x = np.zeros((N, N_grid))
right_bool_y = np.zeros((N, N_grid))

for j, lo in enumerate(grid_left):
    left_bool_x[:, j] = X[:, i+1] > lo
    left_bool_y[:, j] = Y[:, i+1] > lo
for j, hi in enumerate(grid_right):
    right_bool_x[:, j] = X[:, i+1] < hi
    right_bool_y[:, j] = Y[:, i+1] < hi

collision_bool_x = np.logical_and(left_bool_x,
                                  right_bool_x)
collision_bool_y = np.logical_and(left_bool_y,
                                  right_bool_y)

collision_whr = np.where((np.any(collision_bool_x,
                                 axis=1) &
                          np.any(collision_bool_y,
                                 axis=1)))

# Pick out particles that collided
```

```
coll_id = collision_whr[0]
n_coll = len(coll_id)
x_prev = X[coll_id, i]
y_prev = Y[coll_id, i]
x_now = X[coll_id, i+1]
y_now = Y[coll_id, i+1]
coll_x = collision_bool_x[coll_id]
coll_y = collision_bool_y[coll_id]
grid_l = np.repeat(np.expand_dims(grid_left, axis=0),
                   n_coll, axis=0)
grid_r = np.repeat(np.expand_dims(grid_right, axis=0),
                   n_coll, axis=0)

# Pick out the specific block that each particle collided with
x_wall_l = grid_l[coll_x]
x_wall_r = grid_r[coll_x]
y_wall_l = grid_l[coll_y]
y_wall_r = grid_r[coll_y]

# Check if particle passed each wall of that block
xl_passed = x_prev < x_wall_l
xr_passed = x_prev > x_wall_r
yl_passed = y_prev < y_wall_l
yr_passed = y_prev > y_wall_r

# Apply bounceback to the walls that particle passed
x_now[xl_passed] -= 2*(x_now[xl_passed] - x_wall_l[xl_passed])
x_now[xr_passed] -= 2*(x_now[xr_passed] - x_wall_r[xr_passed])
y_now[yl_passed] -= 2*(y_now[yl_passed] - y_wall_l[yl_passed])
y_now[yr_passed] -= 2*(y_now[yr_passed] - y_wall_r[yr_passed])

# Insert back to main array
X[coll_id, i+1] = x_now
Y[coll_id, i+1] = y_now
```