# Random Walks with Reinforcement: If a drunk bird kind of knows where it's going, does it still get lost forever?

Final Report
M375T - Experimental Math, Spring 2025

Carrie Liang (sl54435)

April 28, 2025

## 1 Background

> "A drunk man will find his way home, but a drunk bird may get lost forever."
>
> – *Shizuo Kakutani*

**Stochastic processes** are collections of **random variables**, indexed by "time" either discretely or continuously. For example, a single coin toss can be considered a random variable. A gambler who plays a game tossing a coin infinitely many times is a discrete-time stochastic process, since we can model this scenario by a whole *sequence* of random variables.

This is the classic example of the **simple symmetric random walk** in one dimension. Imagine a particle on a number line that starts at the origin, position 0. Each time a fair coin is tossed, if it lands up on heads, then the particle moves right by 1. Likewise, if it lands on tails, the particle moves left by 1 (subtracting 1 to its current position).

In more precise mathematical language, let $X_0, X_1, X_2, X_3, ...$ be a sequence of random variables such that

$$X_0 = 0, \text{ and } \delta_i = X_i - X_{i-1} = \begin{cases} 1 & \text{with } p = \frac{1}{2} \\ -1 & \text{with } p = \frac{1}{2} \end{cases}$$

The sequence $\{X_i\}$ is the simple symmetric random walk on $\mathbb{Z}$. Each $X_i$ represents the particle's position at time $i$. $\delta_i$ are the steps the particle takes with specified **transition probabilities**, from the current state to another. This is the basic structure of what we study in the rest of this project.

Naturally, we may wonder about the long-run behavior of such stochastic processes. In fact, George Pólya proved that for dimensions $d \leq 2$, the simple random walk is recurrent (meaning that it eventually returns to the origin, given a long enough time. It actually does so infinitely many times, since the random walk "restarts" upon hitting the origin), but for $d > 3$, it is divergent (meaning that it is not guaranteed to return to the origin). [1] Mathematicians later showed that the probability of return in three dimensions is actually around just 0.34.

This famous result is one of the author's favorite math facts, and the inspiration behind this project.

[1] This is what Shizuo Kakutani's quote at the beginning of the report refers to!

# 2 Problem

This project aims to investigate and quantify answers to various questions surrounding the recurrence and divergence of random walks, when the setup of the simple symmetric random walk is changed slightly. We specifically study random walks with **reinforcement**—instead of having the transition probabilities stay constant, they are updated based on the direction in which the particle goes.

## 2.1 Formulation

Let the transition probabilities be $\{p_i\}_{i=1}^n$ where $n$ is the number of possible directions to take and $p_i$ is the probability of going in direction $i$. Here, we are focusing on 1D and 2D, so for 1D, there are two directions, and for 2D, there are four directions. Across all possible directions to go, the transition probabilities must add up to 1, so $\sum_{i=1}^n p_i = 1$.

Additionally, let $k$ be a multiplier that represents the strength of reinforcement.

In the case of **positive reinforcement**, going in a specific direction makes the particle *more* likely to go in that direction again. Transition probabilities for 1D are updated according to the following procedure:

1. Initialize `left` = `right` = 1 and `total = left + right`. Initialize probability of going left or right $p_{\text{left}} = p_{\text{right}} = \frac{1}{2}$.
2. When the particle takes a step at time $i$ according to the current transition probabilities, if it is -1 (left), increment `left` by $k$. If it is +1 (right), increment `right` by $k$. Increment `total`, the total number of steps, by $k$ as well. (This extends to 2D in the same way.)
3. Now update transition probabilities so that $p_{\text{left}} = \frac{\texttt{left}}{\texttt{total}}$ and $p_{\text{right}} = \frac{\texttt{right}}{\texttt{total}}$.
4. Repeat for remaining steps.

In the case of **negative reinforcement**, going in a specific direction makes the particle *less* likely to go in that direction again. Transition probabilities for 1D are updated according to the following procedure:

1. Initialize `left` = `right` = 1 and `total = left + right`. Initialize probability of going left or right $p_{\text{left}} = p_{\text{right}} = \frac{1}{2}$.
2. When the particle takes a step at time $i$ according to the current transition probabilities, if it is -1 (left), increment `right` by $k$. If it is +1 (right), increment `left` by $k$. Increment `total`, the total number of steps, by $k$ as well. (For 2D, add $k$ to each direction that is not the direction just taken, and increment `total` by $3k$.)
3. Now update transition probabilities so that $p_{\text{left}} = \frac{\texttt{left}}{\texttt{total}}$ and $p_{\text{right}} = \frac{\texttt{right}}{\texttt{total}}$.
4. Repeat for remaining steps.

Setting $k = 1$ is the standard reinforcement strength. As $k$ gets smaller, the effect of reinforcement is weakened, and as $k$ gets bigger, the effect of reinforcement is strengthened.

## 2.2 Examples

This graph shows a sample trajectory for each of the three scenarios in 1D, intended to provide intuition for how an average positively/negatively reinforced and normal trajectory compare to each other.

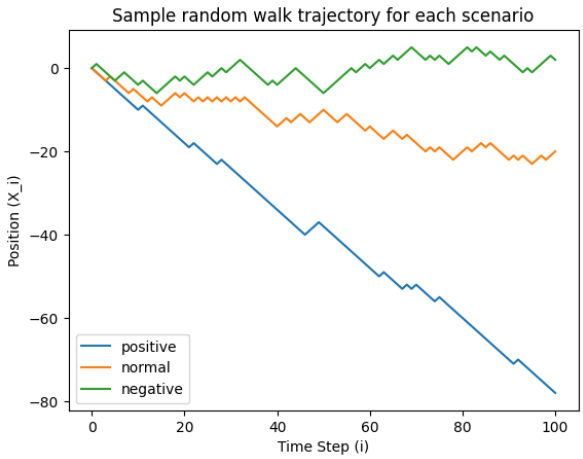



Figure 1: A single sample trajectory for each scenario, color coded by the type. For each time step $i$ on the x-axis, we plot the position $X_i$ on the y-axis.

We see that the positively reinforced trajectory consistently goes to more and more negative values, while the negatively reinforced trajectory hovers around 0. The normal random walk in general went to more negative positions, but switches between directions much more often compared to the positively reinforced case. These are just examples of one possible trajectory, but with very contrasting behaviors; we will see soon what happens to the whole *collection* of possible paths.

## 2.3 Research Questions

Here are the main themes this report revolves around. I will attempt to answer each of these and compare numerical results for the three scenarios (normal, positive reinforcement, negative reinforcement):

1. **What is the distribution of particle locations after $n$ steps? How does variance in particle locations evolve with time?**
2. **What is the distribution of the number of steps it takes for a walk to return home? What is the probability of returning to the origin within $n$ steps? Does this look like it will converge to a particular value as $n \to \infty$?**
3. **What is the distribution of the number of returns to the origin? What are features of walks for each case that make them eventually revisit the origin?**
4. **How do these results compare in one dimension and two dimensions?**
5. **How does the strength of reinforcement $k$ impact answers to the above questions?** [2]

[2] At the time of submission, I only had time to get to this for the first question for 1D.

# 3 Significance and Related Areas

Stochastic processes are used to model many real-world phenomena, from stock prices to population dynamics. By tacking on different properties and constraints to the simple concept of a random process, just as what the laws of physics or psychology dictate, we can expand the selection of problems that we can solve with the help of stochastic processes.

Specifically, the reinforcement feature at the core of this investigation comes up in other areas of probability with models such as Polya's urn and the Chinese Restaurant Process. In those models, similar to how reinforcement is defined in this project, the probability of selecting an object is defined to be proportional to the number of times it has already been selected. Another related subject is random graphs with preferential attachment, a mechanism in which vertices already linked to many others have a higher probability of receiving new connections. For example, famous celebrities are likely to receive even more connections on social media, and already influential research papers are more likely to be read and cited.

Through this project, by exploring the basics of how random walks with reinforcement behave, I hope to build intuition for how they contrast with basic simple symmetric random walks and begin to understand the larger implications for more complex mathematical models where reinforcement arises. Personally, probability is my favorite area in mathematics because many of the constructions relate to our experiences in the real world and are thus easily understandable, but oftentimes, solutions defy expectations and shed light on much deeper and beautiful mathematical theories.

# 4 Methods

All experiments used the Python programming language, with source code in the References.

I anticipated that an effective way to gather the statistics I needed to answer my research questions would be to generate a large number of random walks for a large amount of time steps. This would give me a table of where each sampled random walk was located at each time step. Looking at a single *row* gives a single trajectory, while looking at a single *column* gives information about where a bunch of trajectories behaved after that many time steps.

To work towards this, I first wrote functions to generate a *single* random walk trajectory for each scenario. Then, I had a single function that could apply those single-trajectory functions multiple times (for a *large number* of sample trajectories), to return the table of data I desired.

After this step, all that was needed was essentially more functions that interact with the data in certain ways to get the specific statistics I wanted.

The `numpy` and `pandas` packages were used for generating and storing random walk data, the scipy package was used for fitting distribution curves to the data, and the `matplotlib` and `seaborn` packages were used for creating visualizations of the data. [3]

[3] I acknowledge the use of generative AI (ChatGPT and Google Gemini) for assistance in optimizing my code for data visualization and document formatting.

# 5 Results and Discussion

As a sneak peek, here are 15 sample trajectories for the three cases in one dimension.

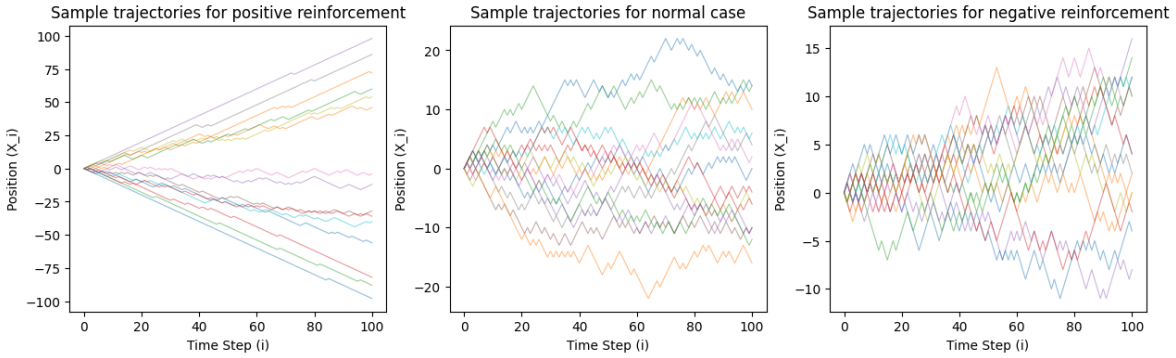



Figure 2: 15 sample trajectories for each case ($X_i$ plotted against $i$). Each color is a different trajectory.

Immediately we see that trajectories of positive reinforcement look very much like straight lines, while the normal and negative cases are more stochastic. Normal and negative reinforcement seem to produce similar looking trajectories, but negative reinforcement tends to produce trajectories that stay closer to the origin, as the y-axis only ranges from -10 to 15 (compared to -20 to 20 for normal).

## 5.1 Distribution of particle locations, 1D

For each case, I first plotted the distribution of particle locations at $X_{10}, X_{100}$, and $X_{1000}$ as histograms.

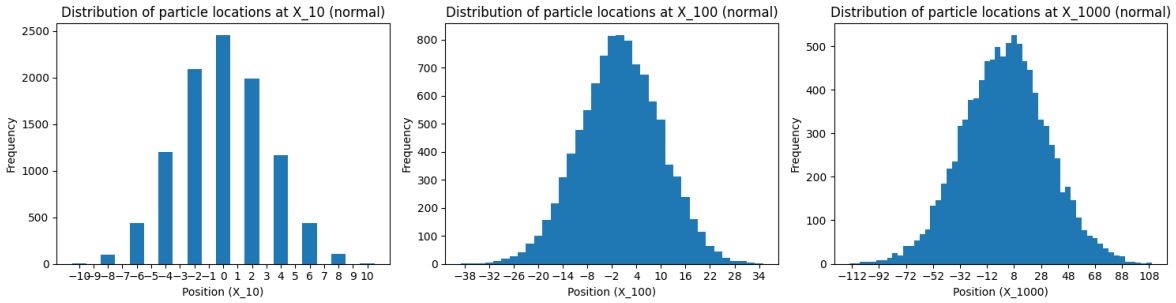



Figure 3: Distribution of particle locations for the normal symmetric case at $X_{10}, X_{100}$, and $X_{1000}$.

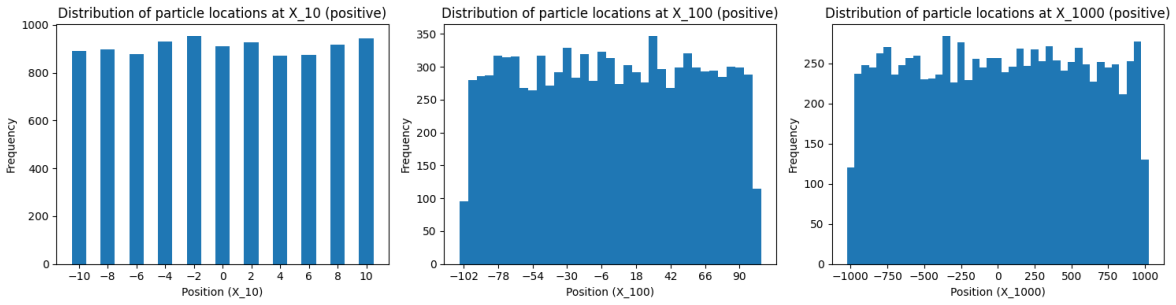



Figure 4: Distribution of particle locations for the positive reinforcement case at $X_{10}, X_{100}$, and $X_{1000}$.

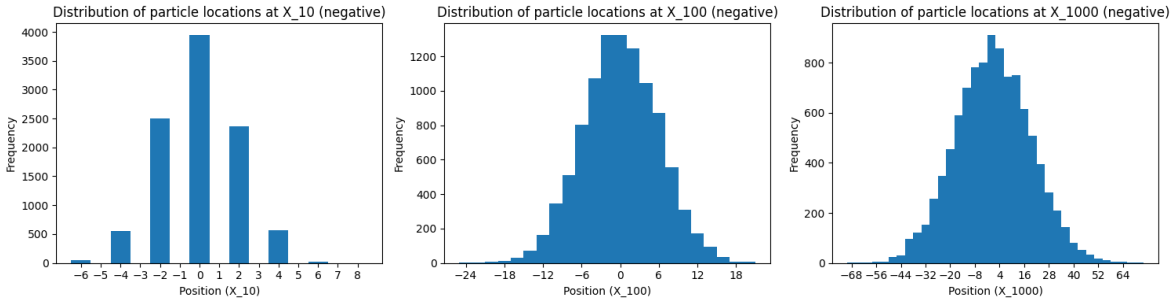



Figure 5: Distribution of particle locations for the negative reinforcement case at $X_{10}, X_{100}$, and $X_{1000}$.

The distributions for the normal and negative reinforcement cases seem to be Gaussian. The distribution for the positive reinforcement cases seems to be uniform. (Side note: The noticeable gaps in the histograms for $X_{10}$ are due to the special property of random walks that their location's **parity** must be the same as the time step's parity: if the particle took an even number of steps, then it can only land on an even location by having both number of steps to the right and to the left be odd/even. Subtracting total number of steps to the left from total number of steps to the right gives a particle's position at a specific time.)

Now back to the question—intuitively, why might we be seeing these distributions?

- Positive reinforcement boosts the chance of going in a particular direction, so it will quickly similar to a **biased** random walk after some steps, when transition probabilities settle to be uneven for going up versus down.
- Negative reinforcement boosts the chance of going in the other direction, so in the long run, the transition probabilities should eventually settle to be close to $\frac{1}{2}$ for each direction, after which it will behave exactly like a normal symmetric random walk.

To quantify the distributions, I plotted the standard deviation (SD) of the particle locations across time.

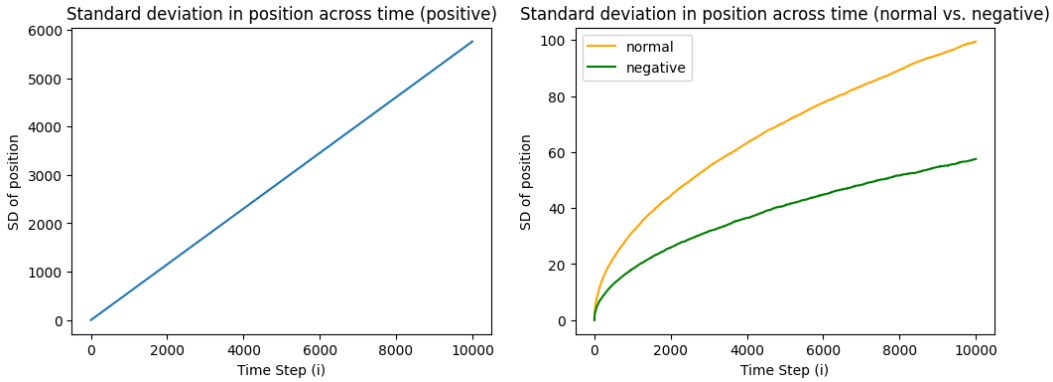



Figure 6: Comparison of standard deviation in position across time. We plot SD of position against the time $i$, and separate the positive case from the normal/negative case since the scales are so different.

The guess for positive reinforcement is that particle positions follow approximately a uniform distribution $X_n \sim U(-n, n)$ (with a slight caveat that is the parity issue mentioned above). For such uniform distributions, the standard deviation $\sigma = \frac{2n}{\sqrt{12}} \approx 0.5773n$. Linear regression for positive reinforcement gives a slope of about 0.5756, which matches expectation!

In the normal case, since probabilities are constant, to derive the formula for how standard deviation of position scales with time step $i$, we can use the formula for variance: $\text{Var}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2$.

- $\mathbb{E}[X] = 0$ since in the long run, we expect a random walk to take about the same number of steps right as steps left. Then $\mathbb{E}[X]^2 = 0$.
- Recalling that a random walk's position $X_i$ is just the sum of its history of increments $\sum_i \delta_i$, we have $\mathbb{E}[X^2] = \mathbb{E}[\sum_i \sum_j \delta_i \delta_j]$. When $i = j$, we have $\delta_i \delta_j = 1$, and when $i \neq j$, we have $\mathbb{E}[\delta_i \delta_j] = 0$ since the increments $\delta_i, \delta_j$ are independent. Therefore $\mathbb{E}[X]^2 = n$, and $\text{Var}(X) = n$ (where $n$ is the number of time steps).
- Finally, $\text{SD}(X) = \sqrt{\text{Var}(X)} = \sqrt{n}$.

We do see behavior that looks like a multiple of $\sqrt{n}$ for both the normal and negative cases, but by what factor do they differ? I used the power law method to calculate the coefficients for the negative reinforcement case by plotting the natural logarithms of the data, assuming that it follows a power law relationship $f(x) = ax^p$. The result was approximately 0.4974 for $p$ and -0.5272 for $\ln(a)$, which corresponds to $a \approx e^{-0.5272} \approx 0.5903$. This seems a significant level above the natural estimate of $\frac{1}{2}$ we might make eyeballing the graph, but I am not too sure where the extra 0.09 may come from.

Next, I investigated how the strength of reinforcement impacts the standard deviation of particle positions at each time step. Since the standard deviation is now a function of two variables (reinforcement strength, time), I visualized the results with a heatmap instead. The darker the gradient, the higher the standard deviation. Note that while the range of colors is the same for both cases, their scales are very different: positive reinforcement ranges from 0 to 1600 while negative reinforcement ranges from 0 to only 25.

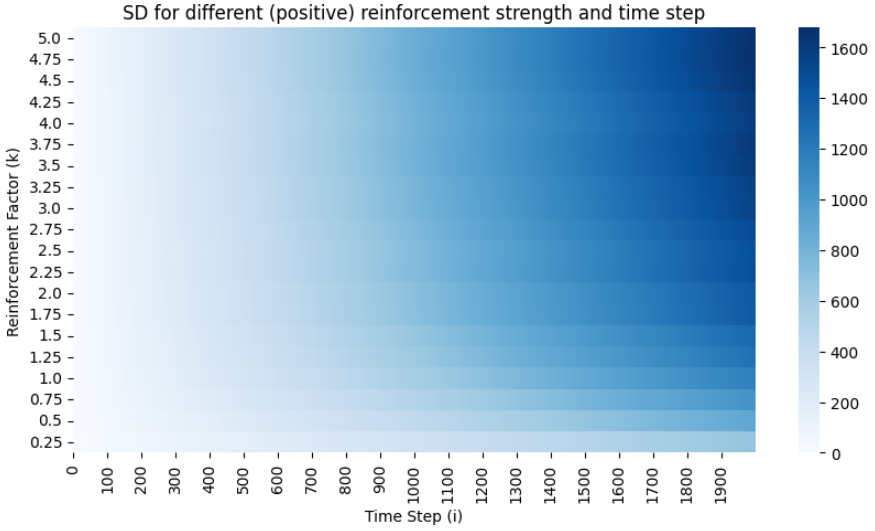



Figure 7: Heatmap of reinforcement factor $k$ and time step $i$'s influence on the standard deviation of position, for positive reinforcement.

We can see a slight curvature in this plot for positive reinforcement, that the standard deviation increases much less as a function of time for a specified reinforcement factor, and especially as that factor drops below 1.

For negative reinforcement (below), the reinforcement factor does not seem to have any effect on how the standard deviation of position increases as a function of time. This isn't too surprising; for positive reinforcement, the strength will only serve to push trajectories further away from the origin,

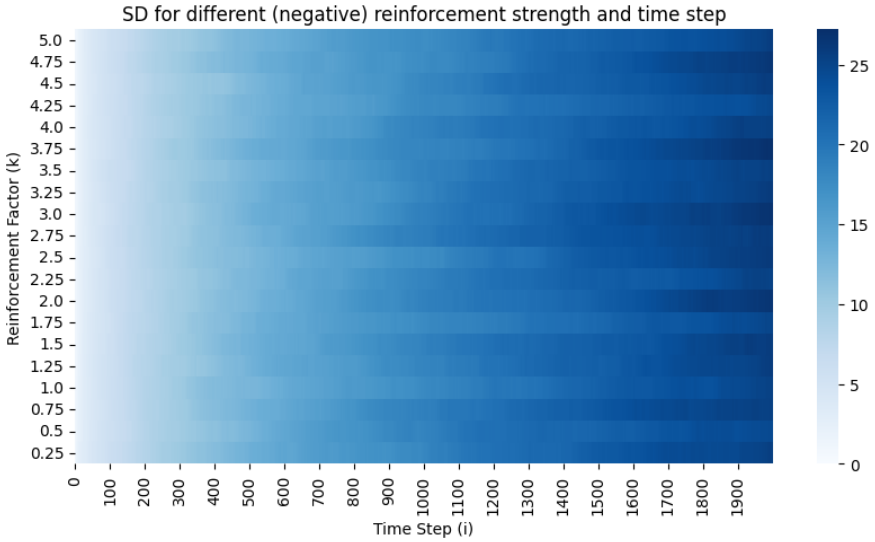



Figure 8: Heatmap of reinforcement factor $k$ and time step $i$'s influence on the standard deviation of position, for negative reinforcement.

while for negative reinforcement, and as we saw before, the negative reinforcement just makes these trajectories behave like basic symmetric random walks.

## 5.2 Probability of returning to the origin, 1D

We finally move on to the central question of this investigation, the probability of returning to the origin. First, I plotted the number of steps to return to the origin for the first time (which is equivalent to the first time index after 0 at which the walk trajectory hits 0). Note that the x-axis is on a natural log scale.

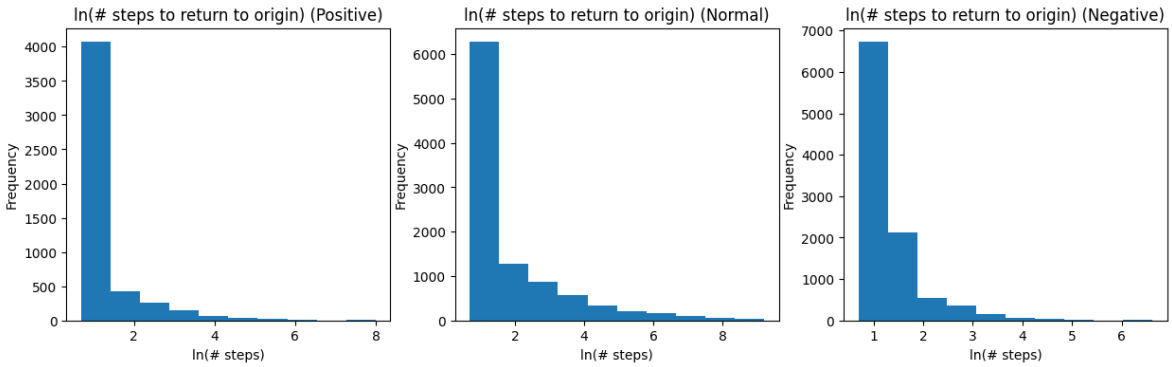



Figure 9: Histograms for number of steps to first return to origin in 1D.

For each scenario, from my 10,000 trial runs with 10,000 steps each, I found the following return probabilities (i.e. the number of walks which returned to the origin within 10,000 steps):

| Positive | Normal | Negative |
|---|---|---|
| 0.5054 | 0.9970 | 0.9999 |

What this data suggests is that for positive reinforcement, out of the walks that do make it back to the origin, the vast majority make it back fairly quickly, within about 10 steps. The distributions are similar for the normal and negative reinforcement cases as well (like the right half of a Gaussian), with the normal case having a longer tail and negative reinforcement having a smaller tail and more volume. For these two cases, all walks eventually return to the origin but negatively reinforced walks do so quicker, as expected.

Here we plot the proportion of walks which have returned by each time step $i$.

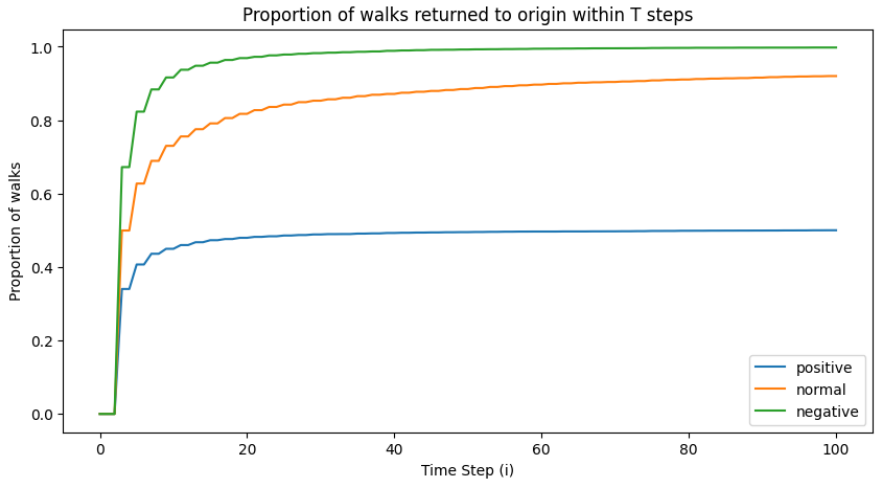



Figure 10: Proportion of walks which have returned to the origin by each time step $i$, by random walk type, in one dimension.

Normal and negative reinforcement seem to approach 1, while positive tends to somewhere around 0.5—around half of the walks end up diverging, and it would be already quite apparent after 50 steps or so based on this graph. Another power law analysis revealed the following coefficients if these three curves are modeled by a function of the form $f(x) = ax^p$:

| | $\ln(a)$ | $a$ | $p$ |
|---|---|---|---|
| Positive | -0.9698 | 0.3791 | 0.0662 |
| Normal | -0.6227 | 0.5364 | 0.1256 |
| Negative | -0.2572 | 0.7731 | 0.0619 |

This means that with reinforcement, the return probability, as a function of the time step, is quicker to converge. As a warning, the results of the power law analysis should only be good at modeling the first hundred or so steps though, since we can guarantee $f(x)$ reaches 1 when $x = \frac{1}{a}^{\frac{1}{p}}$. This cannot be the case with positive reinforcement though by our earlier analysis of how a substantial portion of trajectories' limiting behavior looks like biased random walks (which are *not* guaranteed to return to the origin, unlike symmetric random walks [4]).

[4] An argument for this fact via Math StackExchange: `https://math.stackexchange.com/questions/153123/hitting-probability-of-biased-random-walk-on-the-integer-line`

## 5.3 Number of returns to the origin, 1D

The last major theme of the investigation is the number of returns to the origin that a random walk will have on average. Recall that Polya's result states that for dimensions 1 and 2, the symmetric random walk is recurrent—it returns to the origin infinitely many times, given enough time. However, we cannot actually simulate an infinite time horizon, so an alternative question we can investigate is within a finite number of time steps, what the distribution of the number of returns to the origin is.

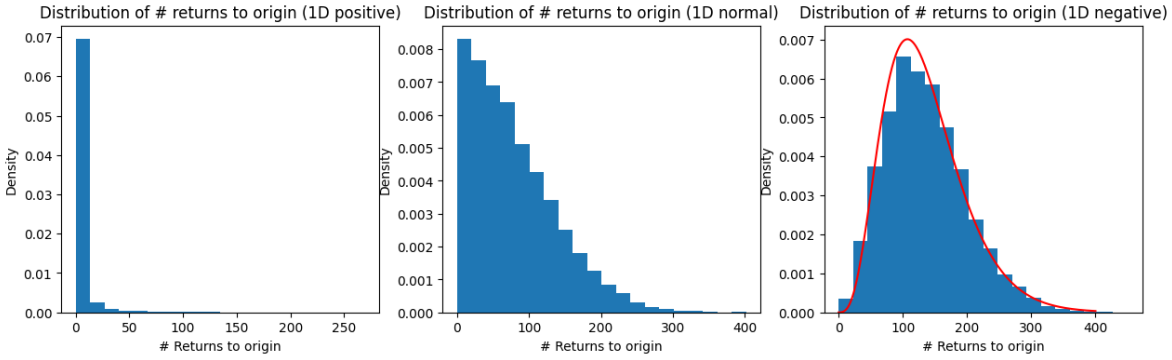
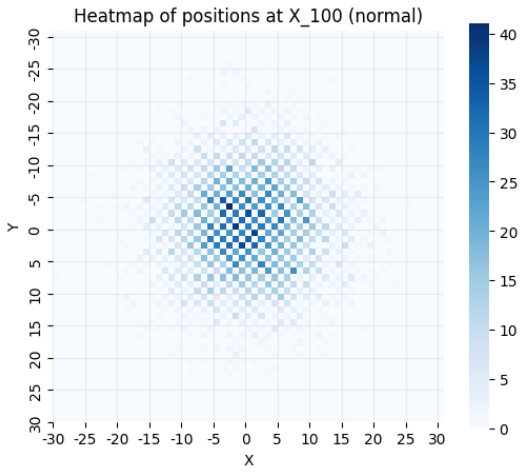



Figure 11: Density plot of distribution of number of returns to origin, for each case. The red line for the normal and negative reinforcement cases is the fitted negative binomial distribution curve.

This is perhaps the most contrasting behavior we've seen so far among the three scenarios! In the 1D case, if a walk does make it back to the origin, it will make it back very few times compared to the other cases. For the normal case, we see a distribution that looks like the right half of a Gaussian. For the negative reinforcement case, the peak is *not near zero*! The median is 129 with a right skew (mean is 135.5 and standard deviation 62.5), indicating it is rare to revisit the origin very many times *and* very few times. It matches pretty well with a **negative binomial distribution** fit (here, we estimate the distribution as $\text{NB}(n, p)$ with $n = \frac{\mu^2}{\sigma^2 - \mu} \approx 4.8661$ and $p = \frac{\mu}{\sigma^2} \approx 0.03467$). It is not immediately clear to me however why these numbers link to the setup, or why a negative binomial distribution works—another possibility is that this follows the gamma distribution instead.

## 5.4 Distribution of particle locations, 2D

We repeat the above analysis for the two-dimensional case, beginning with a plot of where particles are distributed after 100 steps, $X_{100}$. For these 2D questions, I ran 5,000 trials of 10,000 steps each.




The distribution for the positive reinforcement case looks again uniform (note that the gradient for the heatmap only ranges from 0 to 6, so with some room for variance, we can still say it is approximately uniform. It also forms something closer to a diamond shape compared to the 2D negative reinforcement and normal cases, which look more circular (but I can't really tell, it could just be me). Note that the gradient for 2D negative ranges from 0 to 60, while the gradient for 2D normal ranges from 0 to 40 and has slightly more spread in position. Analogously to the 1D case, the distribution of positions for the negative reinforcement resembles that of the normal case but has less standard deviation.

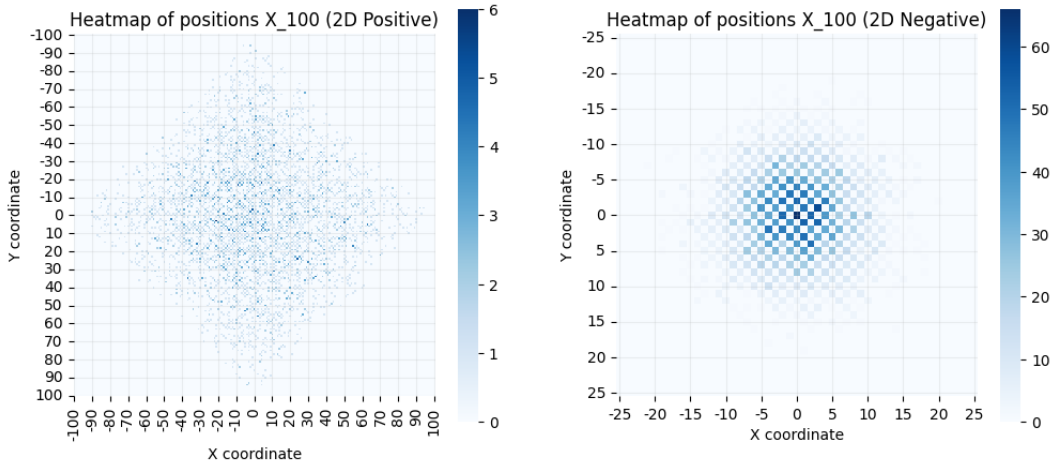



Figure 12: Particle locations at $X_{100}$ in 2D, represented as heatmaps (darker color from the gradient = more trajectories are here at $X_{100}$).

Again, let's check how the standard deviation in position evolves across time. (Actually, I've measured position here as Euclidean distance to the origin.)

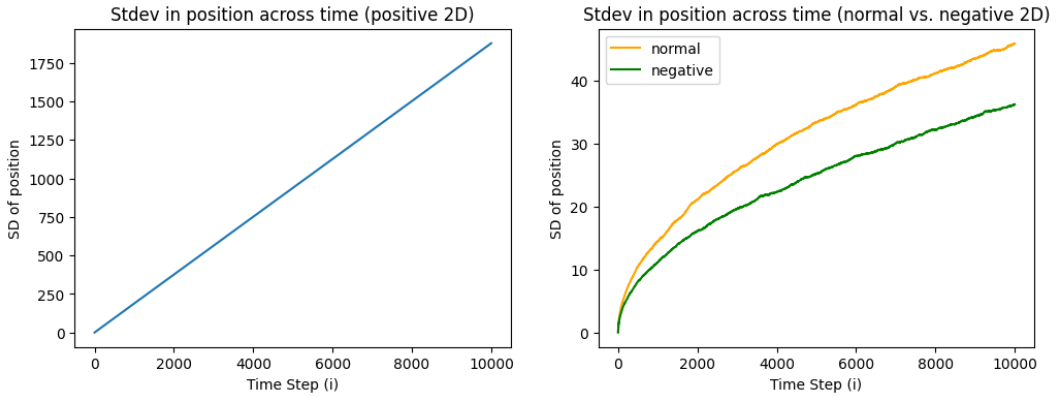



Figure 13: Comparison of standard deviation in position across time. We plot SD of position against the time $i$ as in Figure 6, but now for 2D.

The shapes are the same as for 1D. Power law (so useful!) tells us that the standard deviation in position across time for the normal case and negative reinforcement are indeed still $\sqrt{n}$ relationships. Positive reinforcement again has a uniform distribution. This time, however, the scaling coefficient for negative reinforcement is about 0.7619 of that for the normal case. This indicates that adding these extra degrees of freedom cause negatively reinforced random walks' behavior to converge to that of the normal symmetric random walk more quickly.

## 5.5 Probability of return, 2D

The histograms for the natural logarithm of the number of steps it takes to first return to the origin now look much more similar in size for the normal and negative reinforcement cases in 2D, but with possibly still more mass in the tail for the normal case. The tail area has increased for the 2D positive case, indicating that out of the trajectories which do return, it on average takes more steps to return.

This "spreading out" behavior is expected since in two dimensions, with more choices, it would be harder to choose the necessary ones to go back to the origin. It is not required to cross exactly the origin to get to a certain quadrant, unlike on a 1D number line.

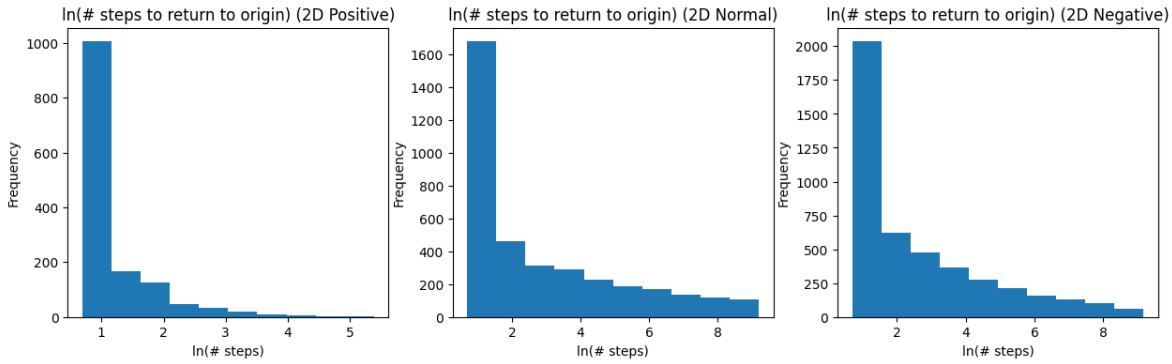



Figure 14: Histograms for number of steps to first return to origin in 2D.

For each scenario, from my 5,000 trial runs with 10,000 steps each, I found the following return probabilities (i.e. the number of walks which returned to the origin within 10,000 steps):

| Positive | Normal | Negative |
|---|---|---|
| 0.2836 | 0.7388 | 0.8882 |

These are substantially lower than for 1D, and let's see how it changes over time:

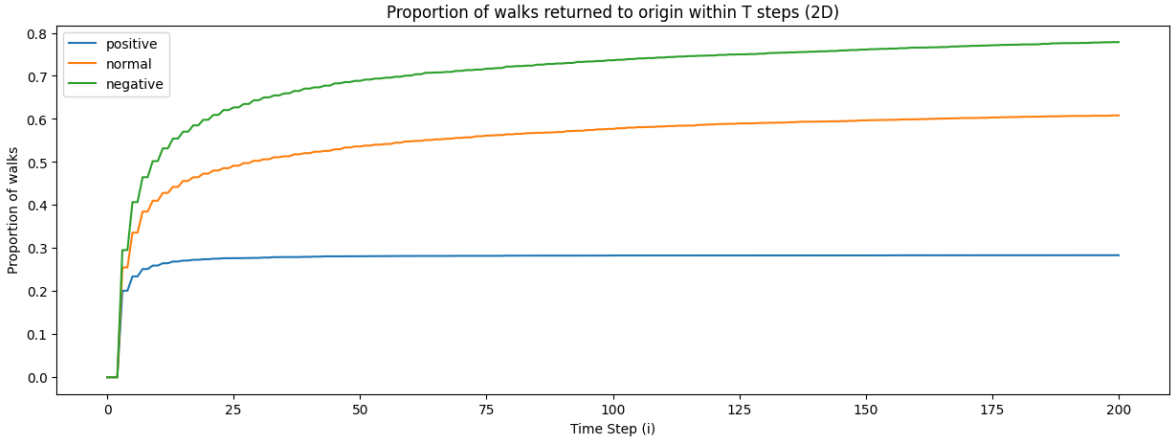



Figure 15: Proportion of walks which have returned to the origin by each time step $i$, by random walk type, in two dimensions.

Positively reinforced trajectories now get even more lost in 2D; the return probability has dropped to around 1/2 of approximately 0.5 for 1D. For two dimensions, within 100 steps, return probabilities for negative reinforcement and normal symmetric walk cases are much slower to converge compared to 1D, and are still nowhere near 1 by double the amount of steps (200).

Another power law analysis gave the following constants for modeling proportion of returned walks as $f(x) = ax^p$:

| | $\ln(a)$ | $a$ | $p$ |
|---|---|---|---|
| Positive | -1.4299 | 0.2393 | 0.0355 |
| Normal | -1.2107 | 0.2979 | 0.1417 |
| Negative | -1.0136 | 0.3628 | 0.1520 |

See Figure 10 for results from the 1D case. Compared to the 1D case, for the 2D case, the coefficient $a$ has decreased while the $p$ has increased (except for positive reinforcement). With larger $p$, the curve "flattens out" (even though it doesn't really since the limit is infinity as $x \to \infty$) at a slower rate. Smaller $a$ serves to squish down the curves a little, but $p$ seems more important here in controlling the shape of the curve.

| | $a_{1D}$ | $a_{2D}$ | $a_{2D}/a_{1D}$ | $p_{1D}$ | $p_{2D}$ | $p_{2D}/p_{1D}$ |
|---|---|---|---|---|---|---|
| Positive | 0.3791 | 0.2393 | 0.6312 | 0.0355 | 0.0662 | 1.8647 |
| Normal | 0.5364 | 0.2979 | 0.5553 | 0.1417 | 0.1256 | 0.8863 |
| Negative | 0.7731 | 0.3628 | 0.4692 | 0.1520 | 0.0619 | 0.4072 |

Table 1: Ratio of fitted power law constants in 2D vs 1D.

Again, this is just an estimate for what this initial portion of the return probability curve looks like for the three scenarios; we know from proofs out there that trajectories in the normal case (and negative reinforcement) do all eventually return to (0,0), though it may take a very long time.

## 5.6 Number of returns to the origin, 2D

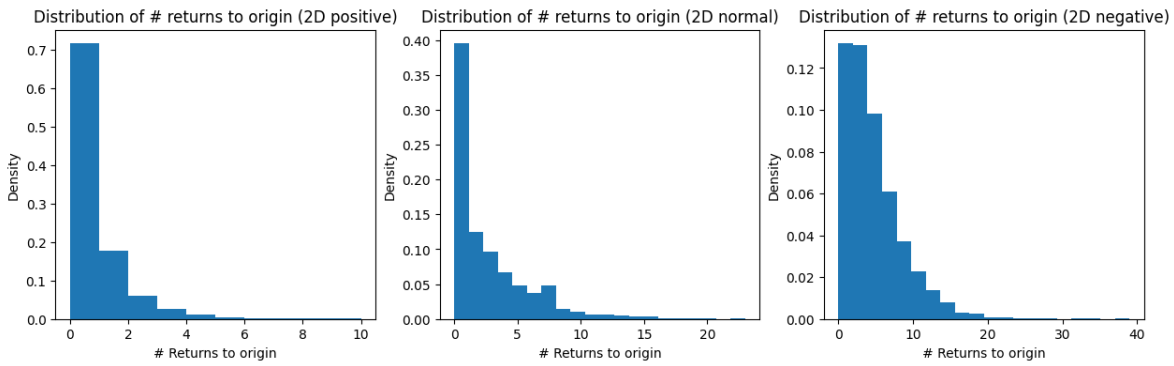



Figure 16: Density plot of distribution of number of returns to origin, for each case, for 2D.

In comparison to Figure 11, we no longer see the shape reminiscent of a negative binomial distribution that peaks at a nonzero value for the negative reinforcement case—instead, it has a half-bell curve shape. The distributions now, roughly speaking, peak at 0 and sharply drop in an exponential manner afterwards. The 2D normal case especially has less of a noticeable half-bell curve shape compared to before and now looks exponential.

Looking at the distribution of the number of returns is also a way to get a sense of how quickly trajectories tend to revisit the origin (after which the random walk essentially resets, at least in the symmetric case, due to the constant transition probabilities). Here, the plots' scale is much smaller—almost none go above 30 returns within 10,000 steps, but for the 1D case, some walks return hundreds of times.

# 6 Conclusion

The results of my experiments largely aligned with intuition regarding how reinforced walks would behave. To summarize, it is reasonable to expect that in general, positively reinforced random walks would send a portion of trajectories to drift off in one direction. On the other hand, negative reinforcement serves as a restoring force to try preventing walks from drifting too far, although a few still end up escaping. Moreover, with an added dimension, walks may have more freedom to wander off, or be harder to rein in.

The more interesting contribution of this project however was to obtain numbers that concretely described the differences in these scenarios. Although I wasn't able to derive a mathematical formula for them, the power law analysis still provides some insights. The most important takeaways from this, in my opinion, are:

- 2D makes everything more lost! The "bird" which somewhat knows where it's going (with negative reinforcement) is great at navigating in one dimension, but the power is not as great relative to the symmetric case with an extra dimension.
- The contrast in the distribution of the number of returns to the origin is quite staggering, especially the negative binomial-esque distribution for 1D with negative reinforcement. But when we think about it, it does kind of make sense that it's rare for this case where walks are actively pushed back to the origin (at least in the first few steps) to revisit it only a few times.

## 6.1 Further Exploration

To answer the original question in this project's title, I think I would have had to expand the project to 3D since that's where recurrence breaks for symmetric random walks. However, this project was already getting really long, and to repeat the analysis for three dimensions (and get some statistics on return probabilities in the long run), I think it would have taken a substantial amount of computing power that my laptop probably doesn't have. I would consider doing a follow-up to this in the future though when I have a better computer that can handle the extra dimension.

I also intended to do more experiments for the strength of the reinforcement factor, but it also would have taken a long time for code to run to get more results (since essentially I would be running the same experiment for maybe 100 different combinations of $k$ with other parameters, and then compiling those results). I focused on the more basic concepts for this project, but if I were to extend it in the future, I would do more analysis for reinforcement strength for the other questions and in 2D.

When originally proposing this project, I didn't anticipate how many small side quests I would end up undertaking during experimenting and writing, and how much work it would be even for one dimension, especially when I was trying to figure out how the negative binomial distribution might connect to my scenario. There were many ideas I originally had for extending this project but weren't able to get to before this was due. One idea I think would particularly have been interesting was:

> *How does this all play out for named probability distributions?* For example, Brownian motion is a continuous-time stochastic process where the increments for time $s < t$ follow a normal distribution: $B_t - B_s \sim \mathcal{N}(0, t - s)$. Emulating reinforcement here would be shifting the mean $\mu$ right or left depending on the direction just taken.

If I had started earlier, I also would have tried to include more mathematical analysis and try to derive how we arrive at the particular distributions I saw (especially the negative binomial-esque distribution for 1D number of returns to the origin with negative reinforcement).

### 6.2 Reflection

It was a fulfilling experience to have carried out an independent inquiry from scratch. I learned a lot during this course and while working on this project regarding how to use computer experiments to investigate a question of my choice, and weave the results into a complete story surrounding the topic.

I certainly was too ambitious with the goals I had initially set, but I'm still decently satisfied with the content I was able to create. I got lots of practice using code to visualize my findings and implemented techniques we learned in class like using the power law to quantify two variables' relationship. Special acknowledgments to my M375T Spring 2025 classmates (especially Group 2), who demonstrated some really inspirational ways of tackling problems, and Dr. Sadun for a great semester :)

## 7 References

Link to Jupyter Notebook source code (hosted on GitHub): `https://github.com/cliang508/m375t-rrw`

`https://mathworld.wolfram.com/PolyasRandomWalkConstants.html`

`https://mathworld.wolfram.com/NegativeBinomialDistribution.html`

`https://www.cs.cmu.edu/afs/cs/academic/class/15251-f03/Site/Materials/Recitations/recitation08/recitation08.pdf`