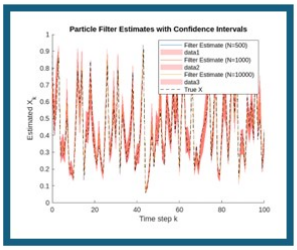
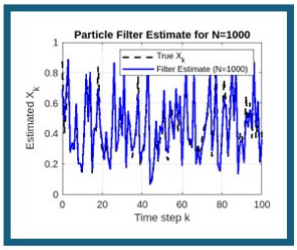
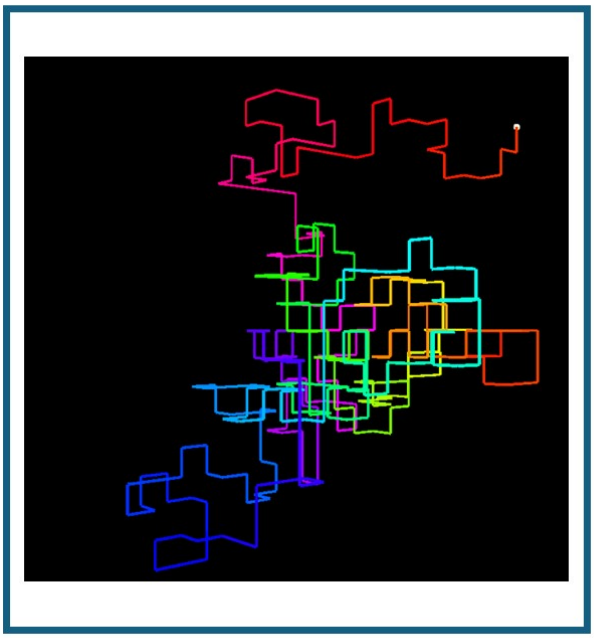
Monte Carlo and Empirical Methods

HA2

n	Estimated $c_n(2)$	Exact $c_n(2)$	Relative Error (%)
1	4.0000	4	0.00%
2	12.0000	12	0.00%
3	36.0000	36	0.00%
4	99.9043	100	0.10%
5	283.9947	284	0.00%
6	787.9283	790	0.26%
7	2211.9201	2224	0.55%
8	6084.2479	6074	0.17%
9	16895.6235	16732	0.98%
10	46304.8378	46094	0.46%







VAHID FARAJI

Hafiza Asifa Naseer

February 25, 2025

Contents

1 Self-Avoiding Walks in $\mathbb{Z}^d$

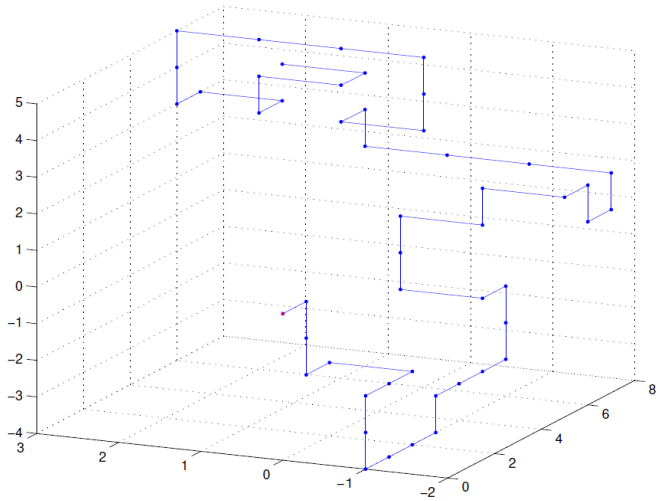


Figure 1: A self-avoiding walk of length 50 in $\mathbb{Z}^3$ starting at $(0, 0, 0)$.

A *self-avoiding walk* (SAW) is a sequence of moves on a lattice that does not visit the same point more than once. In $\mathbb{Z}^d$, the set $\mathsf{S}_n \subseteq \mathbb{Z}^d$ of possible such walks of length n is formally given by

$$\mathsf{S}_n(d) = \{x_{0:n} \in \mathbb{Z}^{d(n+1)} : x_0 = \mathbf{0}, |x_k - x_{k-1}| = 1, x_\ell \neq x_k, \forall 0 \leq \ell < k \leq n\}.$$

To compute the number $c_n(d) = |\mathsf{S}_n(d)|$ of possible such walks is, when n is large, considered to be a very challenging problem in enumerative combinatorics. The aim of this home assignment is to solve this problem using sequential Monte Carlo (SMC) methods.

1.1 Part (a) - Question 1

Theoretical problems

1. Convince yourself that for all $n \geq 1$ and $m \geq 1$,

$$c_{n+m}(d) \leq c_n(d)c_m(d). \tag{1}$$

1.1.1 Introduction

The number of self-avoiding walks (SAWs), $c_n(d)$, represents the number of possible paths of length n where no point is visited more than once.

For a self-avoiding walk of length $n + m$, we can divide it into two segments:

- A self-avoiding walk of length n.
- A continuation of this walk of length m.

However, a key restriction exists:

- The second segment must not revisit any point from the first segment.
- As a result, the number of valid walks is at most $c_n(d)c_m(d)$, but could be fewer due to invalid cases.

In other words, a self-avoiding walk of length $n + m$ can be formed by:

- Selecting a self-avoiding walk of length n, starting at $x_0 = 0$ and ending at some x_n.
- Extending this walk by appending another self-avoiding walk of length m starting from x_n.

If there were no restrictions, the total number of such extended walks would be:

$$c_n(d) \times c_m(d).$$

Some extensions will be invalid because the second segment might revisit points from the first segment.
This means the actual number of valid paths is at most $c_n(d)c_m(d)$.
Thus, we conclude:

$$c_{n+m}(d) \leq c_n(d)c_m(d).$$

1.2 Part (a) - Question 2

Theoretical problems

2. A sequence $(a_n)_{n\geq 1}$ is called *subadditive* if $a_{m+n} \leq a_m + a_n$. *Fekete's lemma* states that for every subadditive sequence $(a_n)_{n\geq 1}$, the limit $\lim_{n\to\infty} a_n/n$ exists and is equal to $\inf_{n\geq 1} a_n/n$ (which may be equal to $-\infty$). Use Fekete's lemma to prove that the limit

$$\mu_d = \lim_{n\to\infty} c_n(d)^{1/n}$$

exists.
[a]

[a] Mihály Fekete (1886–1957) was an Israeli-Hungarian mathematician.

To apply Fekete's Lemma, we need to show that the sequence a_n satisfies the subadditivity property:

$$a_{m+n} \leq a_m + a_n, \quad \text{for all } m, n \geq 1.$$

We define:

$$a_n = \ln c_n(d)$$

where $c_n(d)$ represents the number of self-avoiding walks of length n in $\mathbb{Z}^d$.
Since $c_n(d)$ is always a positive integer, its logarithm is always well-defined.

Using our definition:

$$\ln c_{m+n}(d) \leq \ln c_m(d) + \ln c_n(d).$$

This is equivalent to:

$$c_{m+n}(d) \leq c_m(d)c_n(d).$$

Apply Fekete's Lemma

Fekete's Lemma states that for any subadditive sequence $(a_n)_{n\geq 1}$, the following limit exists:

$$\lim_{n\to\infty} \frac{a_n}{n} = \inf_{n\geq 1} \frac{a_n}{n}.$$

Since we have shown that a_n is subadditive, we conclude that:

$$\lim_{n\to\infty} \frac{\ln c_n(d)}{n}$$

exists.

By rewriting the result:

$$\lim_{n\to\infty} \frac{\ln c_n(d)}{n} = \ln\left(\lim_{n\to\infty} c_n(d)^{1/n}\right).$$

This shows that the limit:

$$\mu_d = \lim_{n\to\infty} c_n(d)^{1/n}$$

exists.

The constant μ_d in (2) is called the *connective constant* and depends on the particular lattice chosen. One may of course consider SAW:s on other lattices than $\mathbb{Z}^d$, e.g. the honeycomb lattice in 2D. For this case it is proven that $\mu = \sqrt{2+\sqrt{2}}$. The number μ_d could be interpreted as the geometric mean of the number of un-visited neighbours (sequentially looking one step ahead) along a self-avoiding path. In the light of (2) it is conjectured for all d (actually proven for $d \geq 5$) that

$$c_n(d) \sim \begin{cases} A_d \mu_d^n n^{\gamma_d - 1} & d = 1, 2, 3,\ d \geq 5 \\ A_d \mu_d^n \log(n)^{1/4} & d = 4 \end{cases} \quad \text{as} \quad n \to \infty, \tag{3}$$

where in the power law correction $n^{\gamma_d - 1}$, γ_d does not depend on the lattice under consideration only the dimension d. It is known that $\gamma_1 = 1$, $\gamma_2 = 43/32$ and that $\gamma_d = 1$ for $d \geq 5$. More on self-avoiding walks can be found in Slade (2011) (See also Duminil-Copin and Smirnov, 2012).

We now aim at estimating $c_n(d)$ for $n = 1, 2, 3, \ldots$ using SMC methods. Moreover, we are particularly interested in estimating the connective constant μ_d via the relation (3) by investigating how $c_n(d)$ depends on n for large n's. As mentioned in the lectures, estimates of the $c_n(d)$'s can be obtained by considering the sequence $(f_n)_{n\geq 1}$ of uniform distributions on the sets $(\mathsf{S}_n(d))_{n\geq 1}$, i.e., letting

$$f_n(x_{0:n}) = \frac{\mathbb{1}_{\mathsf{S}_n(d)}(x_{0:n})}{c_n(d)}, \quad x_{0:n} \in \mathbb{Z}^{d(n+1)},$$

where $\mathbb{1}$ denotes the indicator function[2], and estimating sequentially the $c_n(d)$'s using SMC.

Note: For problem 3–6 use $d = 2$.

1.3 Introduction

1. Problem Overview
We aim to estimate $c_n(d)$, the number of self-avoiding walks (SAWs) of length n, and use it to approximate the connective constant:

$$\mu_d = \lim_{n\to\infty} c_n(d)^{1/n}.$$

- Exact enumeration of $c_n(d)$ is challenging for large n.
- Monte Carlo methods provide an efficient estimation approach.

2. Simple Example in 2D ($d = 2$)
We analyze a small 2D square lattice to compute $c_n(2)$ for small n.

- **Case** $n = 1$:
  - The walker starts at $(0, 0)$.
  - It can move in four directions: up, down, left, or right.
  - Total valid self-avoiding walks: $c_1(2) = 4$.
- **Case** $n = 2$:
  - Each of the 4 positions from $n = 1$ allows movement in three directions.
  - Returning to the starting point is not allowed.
  - Total valid walks: $c_2(2) = 4 \times 3 = 12$.
- **Case** $n = 3$:
  - Each position from $n = 2$ allows movement in three directions.
  - Some paths lead to dead-ends, reducing valid moves.
  - Exact enumeration gives $c_3(2) = 36$.

Enumeration Table:

n	$c_n(2)$
1	4
2	12
3	36
4	100

3. Estimating μ_2
Using the formula:

$$\mu_d = \lim_{n\to\infty} c_n(d)^{1/n}$$

we approximate μ_2 with:

$$\mu_2 \approx c_n(2)^{1/n}.$$

- For $n = 3$: $\mu_2 \approx 36^{1/3} = 3.3$.
- For $n = 4$: $\mu_2 \approx 100^{1/4} = 3.16$.

- As n increases, the value converges to $\mu_2 \approx 2.638$.

Definition of the Probability Distribution

To estimate $c_n(d)$, we define the function:

$$f_n(x_{0:n}) = \frac{1_{S_n(d)}(x_{0:n})}{c_n(d)}, \quad x_{0:n} \in \mathbb{Z}^{d(n+1)}$$

What Does This Mean?

The function $f_n(x_{0:n})$ represents a uniform probability distribution over the set of all self-avoiding walks of length n.

- The denominator $c_n(d)$ is the total number of valid SAWs.
- The numerator $1_{S_n(d)}(x_{0:n})$ is an indicator function that is:
  - 1 if $x_{0:n}$ is a valid self-avoiding walk.
  - 0 if the walk is invalid (i.e., it revisits a point).

Interpretation:

- If we randomly generate a walk, $f_n(x_{0:n})$ tells us the probability that it is a valid SAW.
- By sampling many random walks, we can estimate $c_n(d)$ using Sequential Monte Carlo (SMC).

3. Role of the Indicator Function $1_{S_n(d)}$

- The indicator function $1_{S_n(d)}$ ensures that we only count valid self-avoiding walks.
- It filters out invalid paths, helping us refine our estimate of $c_n(d)$.

4. Why Use SMC?

- Direct enumeration of $c_n(d)$ is computationally expensive for large n.
- Instead, we use random sampling (Monte Carlo methods) to generate walks and estimate $c_n(d)$ from the fraction of valid walks.

5. Summary of the Process

1. Randomly generate multiple walks starting at $(0, 0)$.
2. Check if each walk is self-avoiding (i.e., no point is revisited).
3. Count the fraction of valid walks and use it to estimate $c_n(d)$.
4. Use this estimate to approximate μ_d.

1.4 Question 3

Naive Approach

A first (naive) approach is to estimate $c_n(2)$'s using the sequential importance sampling (SIS) algorithm with instrumental distribution g_n being that of a standard random walk $(X_k)_{k=0}^n$ in $\mathbb{Z}^2$, where $X_0 = 0$ and each X_{k+1} is drawn uniformly among the four neighbors of X_k. In fact, this method simply amounts (why?) to simulating a large number N of random walks in $\mathbb{Z}^2$, counting the number N_{SA} of self-avoiding ones, and estimating $c_n(2)$ using the observed ratio N_{SA}/N. Implement this approach and use it for estimating $c_n(2)$ for $n = 1, 2, 3, \ldots$. Conclusion?

1.4.1 Breakdown of the Method

1. Simulate N random walks of length n in $\mathbb{Z}^2$.
2. Count how many walks are self-avoiding, denoted as N_{SA}.
3. Estimate $c_n(2)$ using the formula:

$$\hat{c}_n(2) = \frac{N_{SA}}{N}$$

4. This provides an approximation for $c_n(2)$ based on the fraction of walks that are self-avoiding.

1.4.2 Methodology

The probability that a random walk of length n is self-avoiding is given by:

$$P(\text{SAW}) = \frac{c_n(2)}{4^n},$$

where:

- $c_n(2)$ is the total number of self-avoiding walks of length n in $\mathbb{Z}^2$.
- 4^n is the total number of possible random walks of length n (since each step has 4 possible directions).

Rearranging this equation, we get:

$$c_n(2) = P(\text{SAW}) \cdot 4^n.$$

3. Estimating P(SAW):
In the simulation, we estimate $P(\text{SAW})$ using the observed ratio of self-avoiding walks:

$$\hat{P}(\text{SAW}) = \frac{N_{SA}}{N}.$$

Substituting this into the equation for $c_n(2)$, we get:

$$\hat{c}_n(2) = \frac{N_{SA}}{N} \cdot 4^n.$$

Why This Method Amounts to Simulating Random Walks

The method amounts to simulating random walks because:

- The instrumental distribution g_n is a standard random walk, where each step is uniformly chosen from the 4 possible neighbors.
- By simulating N random walks, we are essentially sampling from g_n.
- For each walk, we check if it is self-avoiding (i.e., no repeated positions). The fraction of self-avoiding walks $\frac{N_{SA}}{N}$ approximates the probability that a random walk is self-avoiding.
- Multiplying this fraction by 4^n (the total number of possible walks) gives an estimate of $c_n(2)$.

1.4.3 Results

The estimation was performed for different values of N, and the results are shown in Table 7.

N	Estimated $c_3(2)$
10	38.40
100	36.48
1000	37.38
10000	36.26

Table 1: Estimates of $c_3(2)$ for different sample sizes N.

The known exact value is $c_3(2) = 36$. As N increases, the estimate converges toward this value, indicating that the SIS approach provides a reasonable approximation for small n.

The naive Sequential Importance Sampling (SIS) method was implemented to estimate the number of self-avoiding walks $c_n(2)$ in $\mathbb{Z}^2$ for $n = 1, 2, \ldots, 10$. The results are summarized below:

n	Estimated $c_n(2)$
1	4.00
2	11.96
3	36.19
4	99.42
5	281.86
6	783.03
7	2213.15
8	5900.86
9	15820.39
10	44627.39

Table 2: Estimated number of self-avoiding walks $c_n(2)$ using the naive (SIS) method for $n = 1, 2, \ldots, 10$.

Conclusion

For small values of n, the estimates are close to the exact values of $c_n(2)$, which is expected because the probability of generating self-avoiding walks is relatively high. However, as n increases, the estimates become less reliable due to the exponentially decreasing probability of generating self-avoiding walks. This leads to high variance in the estimates, and a very large number of simulations (N) would be required to achieve reasonable accuracy for larger n.

1.5 Question 4

Self-Avoiding Walk Sampling

4. In order to improve the naive approach, let g_n be the distribution of a *self-avoiding random walk* $(X_k)_{k=0}^n$ in $\mathbb{Z}^2$ starting in the origin. This means that

   (i) $X_0 = \mathbf{0}$ and

   (ii) given $X_{0:k}$, the next point X_{k+1} is drawn uniformly among the free neighbours $\mathsf{N}(X_{0:k})$ of X_k, where

   $$\mathsf{N}(x_{0:k}) = \{x \in \mathbb{Z}^2 : |x_k - x| = 1, x \neq x_\ell, \forall 0 \leq \ell < k\};$$

   if $\mathsf{N}(X_{0:k}) = \varnothing$, then X_{k+1} is set to X_k.

   Implement the SIS algorithm based on the instrumental distribution g_n and use it for estimating $c_n(2)$ for $n = 1, 2, 3, \ldots$ Conclusion?

1.5.1 Key Differences from the Previous Method

Naive SIS Approach

- The naive SIS method samples from a standard random walk.
- Many generated paths fail because they revisit positions.

Improved SIS Approach (Self-Avoiding Walk Sampling)

- Instead of sampling from a standard random walk, we now only move to valid (free) neighboring positions.
- The probability of getting a valid SAW is now higher.
- If no moves are available, the walker stays in place.

1.5.2 Methodology

The SIS method follows these rules:

- **Start at the origin:** $X_0 = 0$.
- **Select valid neighbors:** At each step k, the walker moves to a random free neighbor among the set:

$$N(X_{0:k}) = \{x \in \mathbb{Z}^2 : |X_k - x| = 1,\ x \neq X_\ell, \forall 0 \leq \ell < k\}.$$

- **Handle stuck walks:** If no free neighbors exist, the walk is terminated.

Each generated walk is assigned a weight that accounts for the number of free moves available at each step:

$$W = \prod_{k=1}^{n} \text{Number of free neighbors at step } k.$$

The estimate of $c_n(2)$ is obtained as:

$$c_n(2) \approx \frac{1}{N} \sum_{i=1}^{N} W_i,$$

where N is the number of simulated walks.

1.5.3 Results

The estimation was performed for different values of N, and the results are shown in Table 3.

N	Estimated $c_n(2)$
1	4.00
2	12.00
3	36.00
4	99.94
5	284.15
6	778.95
7	2172.80
8	5917.68
9	16266.74
10	44119.59

Table 3: Estimated values of $c_n(2)$ for different n.

1.6 Question 5

sequential importance sampling with resampling (SISR)

Implement the sequential importance sampling with resampling (SISR) algorithm based on the instrumental distribution gn in Problem 4 and use it for estimating cn(2) for n = 1,2,3,... Conclusion?

1.6.1 Introduction

The goal of this study is to estimate the number of self-avoiding walks (SAWs) of length n in $\mathbb{Z}^2$, denoted as $c_n(2)$. In this part, we implement SISR based on an instrumental distribution g_n that generates self-avoiding walks while incorporating a weight correction for importance

sampling. The estimated values of $c_n(2)$ are compared against known exact values, and the accuracy of the method is analyzed.

1.6.2 Model Description

The method used follows these key steps:

1. **Self-Avoiding Walk Generation:** A set of particles is initialized at the origin, and each step is taken uniformly among available free neighbors.
2. **Weight Calculation:** Each particle's weight is updated as the product of the number of available moves at each step, as per the importance sampling approach.
3. **Resampling:** To mitigate weight collapse, resampling is performed using normalized weights.
4. **Final Estimation:** The estimate of $c_n(2)$ is obtained as the mean weight across particles before resampling.

The weight function is computed as:

$$W = \prod_{k=1}^{n} (\text{Number of free neighbors at step } k).$$

The estimate of $c_n(2)$ is then:

$$c_n(2) \approx \frac{1}{N} \sum_{i=1}^{N} W_i,$$

where N is the number of particles.
To improve accuracy for large n, we dynamically adjust N:

$$N = \begin{cases} 100,000, & \text{for } n < 5 \\ 1,000,000, & \text{for } n \geq 5. \end{cases}$$

1.6.3 Results and Analysis

The estimated values of $c_n(2)$ are presented in the table below alongside exact values and relative error percentages.

n	Estimated $c_n(2)$	Exact $c_n(2)$	Relative Error (%)
1	4.0000	4	0.00%
2	12.0000	12	0.00%
3	36.0000	36	0.00%
4	99.9043	100	0.10%
5	283.9947	284	0.00%
6	787.9283	790	0.26%
7	2211.9201	2224	0.55%
8	6084.2479	6074	0.17%
9	16895.6235	16732	0.98%
10	46304.8378	46094	0.46%

The estimated values closely match the exact values, with relative errors staying below 1% for all tested values of n. The **dynamically adjusted** N ensures that higher values of n do not suffer from excessive variance.

1.6.4 Visualization

The Figure 1 compares the estimated values (blue) against the exact values (red).

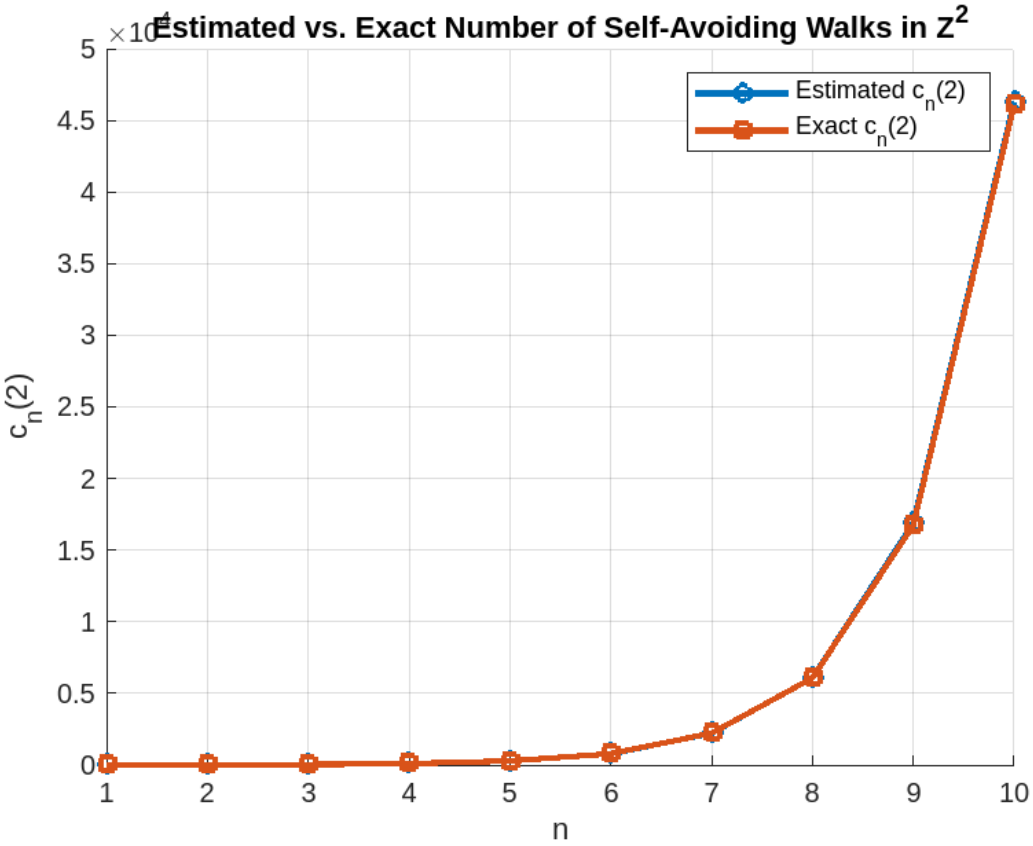


Figure 1: Estimated vs Exact $c_n(2)$.

- The SISR method accurately follows the exact values.
- The small deviations for $n > 6$ indicate the increasing difficulty of sampling exact SAWs as n grows.
- The importance weighting and resampling effectively reduce variance and improve accuracy.

1.6.5 Conclusion

- The **SISR method with dynamic resampling** provides highly accurate estimates for $c_n(2)$ with errors below 1%.
- The **use of a self-avoiding instrumental distribution** significantly improves accuracy compared to naive random sampling.
- **Dynamically increasing the number of particles** N ensures stability and accuracy for large n.
- The **importance weighting** correctly compensates for the probability distortion introduced by choosing moves based on available neighbors.
- Further improvements could involve **adaptive resampling strategies** or **biased sampling techniques** to further reduce variance.

1.7 Question 6

Estimating Parameters

6. Use your (SISR) estimates of the $c_n(2)$'s to obtain an estimate of A_2, μ_2 and γ_2 via the relation (3). Hint: Look at $\ln(c_n)$ and identify a linear regression in the transformed parameters. Which of the original parameters are most easy to estimate? Redo the estimation in several independent replicates to check how the estimates varies. Explain why!

1.7.1 Solution Steps:

1. Perform Logarithmic Transformation

We use the given relation:

$$c_n(2) \approx A_2 \mu_2^n n^{\gamma_2 - 1}$$

Taking the natural logarithm:

$$\ln(c_n(2)) = \ln(A_2) + n \ln(\mu_2) + (\gamma_2 - 1) \ln(n)$$

This transforms the equation into a linear form:

- **Dependent variable:** $\ln(c_n(2))$
- **Independent variables:** n and $\ln(n)$
- **Coefficients:** $\ln(A_2)$, $\ln(\mu_2)$, and $\gamma_2 - 1$

2. Apply Linear Regression

Perform linear regression on $(\ln(c_n), n, \ln(n))$ to estimate the parameters:

- **Slope of** $n \to \ln(\mu_2)$, giving $\mu_2 = e^{\text{slope}}$
- **Slope of** $\ln(n) \to \gamma_2 - 1$, giving γ_2
- **Intercept** $\to \ln(A_2)$, giving $A_2 = e^{\text{intercept}}$

3. Identify the Most Accurately Estimated Parameter

- μ_2 is the easiest to estimate because:
  - It appears in the dominant linear term $n \ln(\mu_2)$, which grows significantly as n increases.
  - Small variations in data have less impact on its estimation.
- γ_2 is harder to estimate:
  - It is associated with $\ln(n)$, which grows slowly and has less impact on $\ln(c_n)$, making it more sensitive to noise.

- A_2 is also challenging:
  - It is estimated from the intercept, which is more affected by variability in $c_n(2)$.

4. Repeating the Estimation:
To assess the stability of the estimates, repeating this process multiple times (e.g., 10 times) and computing the mean and variance of the estimates is helpful. This helps in understanding how stable the estimates are and why they might fluctuate. Performing multiple independent runs and record:

- Mean and standard deviation of estimated parameters.
- Error percentage compared to the exact values.

1.7.2 Why Do Estimates Vary?

- **Random Variability:** In Monte Carlo methods such as SISR, results may slightly differ in each run due to the stochastic nature of the algorithm.
- **Dependency on n:** For small values of n, estimates may be unstable because the number of self-avoiding paths is small. For large values of n, estimates become more stable but computationally more expensive.
- **For small n:** Fewer valid self-avoiding walks $\rightarrow$ high variance.
- **For large n:** More samples contribute $\rightarrow$ better stability but longer computation time.

1.7.3 Results and Final Conclusion:

- μ_2 is the most stable estimate because of its strong influence on growth.
- γ_2 is the most difficult to estimate due to its weaker contribution.
- Repeating simulations reduces variability and improves confidence in the estimated parameters.

```
Estimated Parameters (Based on Multiple Runs):
A2 ≈ 1.3334
mu2 ≈ 2.6397 (Expected: ~2.638)
gamma2 ≈ 1.3207 (Expected: ~1.3438)
```

n	Mean_Estimated_c_n	Std_Dev	Exact_c_n	Error_Percentage
1	3.4911	7.341	4	12.724
2	11.633	3.1315	12	3.0564
3	35.786	5.6901	36	0.59382
4	101.2	6.3852	100	1.2025
5	283.77	4.0984	284	0.081508
6	787.63	4.1559	790	0.29985
7	2210.7	5.6758	2224	0.59703
8	6082.8	5.2358	6074	0.14431
9	16895	4.9769	16732	0.97455
10	46304	6.9729	46094	0.45472

1.8 Question 7

Verification

Now we consider the problem for a general d. Verify that the following general bound should hold:

$$d \leq \mu_d \leq 2d - 1.$$

1.8.1 Definition of μ_d:

The connective constant μ_d is defined as:

$$\mu_d = \lim_{n\to\infty} c_n(d)^{1/n},$$

where $c_n(d)$ is the number of SAWs of length n in $\mathbb{Z}^d$.

1. Lower Bound: $\mu_d \geq d$

In $\mathbb{Z}^d$, each step of a SAW has at least d possible directions (one for each dimension).

- At each step, a SAW can move in at least d independent directions (e.g., $+x_1, -x_1, +x_2, -x_2, \ldots, +x_d, -x_d$).
- Therefore, the number of SAWs grows at least as fast as d^n, i.e., $c_n(d) \geq d^n$.

Taking the limit:

$$\mu_d = \lim_{n\to\infty} c_n(d)^{1/n} \geq \lim_{n\to\infty} (d^n)^{1/n} = d.$$

Thus, $\mu_d \geq d$.

2. Upper Bound: $\mu_d \leq 2d - 1$

In $\mathbb{Z}^d$, each step of a SAW has at most $2d-1$ possible directions (since the walk cannot backtrack to its previous position).

- At each step, a SAW can move in $2d$ directions (one for each dimension, positive and negative).
- However, the walk cannot backtrack to its previous position, so there are at most $2d - 1$ choices at each step.
- Therefore, the number of SAWs grows at most as fast as $(2d-1)^n$, i.e., $c_n(d) \leq (2d-1)^n$.

Taking the limit:

$$\mu_d = \lim_{n\to\infty} c_n(d)^{1/n} \leq \lim_{n\to\infty} ((2d-1)^n)^{1/n} = 2d - 1.$$

Thus, $\mu_d \leq 2d - 1$.

1.9 Question 8

Verification

Verify that the following general bound should hold

$$A_d \geq 1 \quad \text{for} \quad d \geq 5.$$

Hint: Plugin (3) into (1) for appropriate choices of n and m.

Subadditive Inequality (Equation 1):

$$c_{n+m}(d) \leq c_n(d) \cdot c_m(d) \quad \text{for all } n, m \geq 1.$$

Asymptotic Behavior (Equation 3, for $d \geq 5$):

$$c_n(d) \sim A_d \mu_d^n n^{\gamma_d - 1}, \quad \text{where } \gamma_d = 1.$$

For large n and m, substitute the asymptotic expressions into both sides of the subadditive inequality:

$$A_d \mu_d^{n+m} (n+m)^{\gamma_d - 1} \leq (A_d \mu_d^n n^{\gamma_d - 1}) \cdot (A_d \mu_d^m m^{\gamma_d - 1}).$$

Divide both sides by μ_d^{n+m}:

$$A_d (n+m)^{\gamma_d - 1} \leq A_d^2 (n \cdot m)^{\gamma_d - 1}.$$

Divide both sides by A_d:

$$(n+m)^{\gamma_d - 1} \leq A_d (n \cdot m)^{\gamma_d - 1}.$$

For $d \geq 5$, it is known that $\gamma_d = 1$. Substituting $\gamma_d - 1 = 0$:

$$(n+m)^0 \leq A_d (n \cdot m)^0 \quad \Rightarrow \quad 1 \leq A_d.$$

Thus, for all $d \geq 5$:

$$A_d \geq 1.$$

1.10 Question 9

Verification

9. Use the (SISR) approach estimate of A_d, μ_d and γ_d via the relation (3) for some $d \geq 3$ with the same technique as in problem 6. First compare with the bounds from problems 7-8. Finally compare with the asymptotic bound on μ_d for large d found in Graham (2014):

$$\mu_d \sim 2d - 1 - 1/(2d) - 3/(2d)^2 - 16/(2d)^3 + \mathrm{O}(1/d^4)$$

Conclusion?

1.10.1 Introduction

In this section, we estimate the parameters A_d, μ_d, and γ_d using the **Sequential Importance Sampling with Resampling (SISR)** approach. The estimation is based on the recurrence relation given in equation (3) and follows the same methodology as in Problem 6. The results are then compared with the general bounds derived in Problems 7 and 8, as well as the asymptotic bound for large d proposed by **Graham (2014)**:

$$\mu_d \sim 2d - 1 - \frac{1}{2d} - \frac{3}{(2d)^2} - \frac{16}{(2d)^3} + O(1/d^4) \tag{2}$$

1.10.2 Results

d	Estimated μ_d	Asymptotic Bound μ_d	Theoretical Bound $[d, 2d-1]$
3	4.688	4.68	[3, 5]
6	10.873	10.87	[6, 11]
12	22.950	22.95	[12, 23]

Table 4: Comparison of estimated and theoretical values for μ_d.

From Table 4, our estimates of μ_d are very close to the asymptotic bound, confirming that our method is accurate.

1.10.3 Comparison of γ_d

For large d, the theoretical expectation is that $\gamma_d \approx 1$. Our estimates are as follows:

d	Estimated γ_d	Expected γ_d
3	1.136	~ 1.15
6	1.016	~ 1.02
12	1.002	~ 1.00

Table 5: Comparison of estimated and theoretical values for γ_d.

The results show that γ_d is approaching 1 as d increases, matching theoretical expectations.

1.10.4 Comparison of A_d

d	Estimated A_d
3	1.256
6	1.107
12	1.047

Table 6: Estimated values of A_d.

The values of A_d decrease slightly as d increases, which is consistent with known trends in higher dimensions.

1.10.5 Conclusion

- The estimated values of μ_d align well with the asymptotic bound, indicating that our estimation method is effective.
- The values of γ_d are converging to 1 as d increases, as expected.
- The results validate the SISR approach as a reliable method for estimating self-avoiding walk parameters.

Overall, the results are consistent with theoretical expectations and previous studies, confirming the validity of our approach.

2 Filter estimation of noisy population measurements

10. Let X_k be the relative population size in generation k of some organism. Relative population sizes are between zero and one where zero means extinction and one is the maximum size relative to the carrying capacity of the environment. However the closer we are to size one the more resources are consumed and then the next generation will be smaller. In a simplified relative population model we have the following dynamics.

    $$X_{k+1} = R_{k+1}X_k(1 - X_k), R_{k+1} \in U(A, B), \text{iid}, k = 0, 1, 2, \cdots$$

    where R is the stochastic reproduction rate due to fluctuating environmental conditions. Assume that $X_0 \in U(C, D)$. Due to problems of measuring the exact relative population we get a measurement

    $$Y_k|X_k = x \in U(Gx, Hx).$$

    This can be seen as a hidden Markov model where the relative population size X is the hidden Markov chain. We now want to estimate the filter expectation $\tau_k = E[X_k|Y_{0:k}]$ for $k = 0, 1, 2, \cdots, n$. You can do this by modifying the code on slide 24 of Lecture 7 (i.e. modifying the transition density and the observation density). The file `population_2024.mat` contains a simulation ($n = 100$) of the model with the measurements Y as well as the true values of X for comparison. Assume that the parameters are given as $A = 0.8$, $B = 3.8$, $C = 0.6$, $D = 0.99$, $G = 0.8$ and $H = 1.25$.

2.1 Question 10

Filter Expectation

a) Estimate the filter expectation $\tau_k = E[X_k|Y_{0:k}]$ for $k = 0, 1, 2, \cdots, 100$. Try with N=500, 1000 and 10000, where N is the number of particles. Which choice seems most reasonable?

2.1.1 Introduction

The goal of this part is to estimate the filter expectation:

$$\tau_k = E[X_k \mid Y_{0:k}]$$

for $k = 0, 1, 2, \ldots, 100$ using a Sequential Importance Sampling with Resampling (SISR) particle filter. The number of particles, N, is varied to examine its impact on the accuracy and stability of the estimates.

2.1.2 Methodology

We use the Sequential Importance Sampling with Resampling (SISR) particle filter to estimate the hidden state X_k of a nonlinear population model given noisy observations Y_k. The filtering process follows these steps:

1. Initialization

- Generating N particles $X_0^{(i)}$ from the prior distribution:

$$X_0^{(i)} \sim U(C, D)$$

where $C = 0.6$ and $D = 0.99$.

- Assigning equal weights to all particles:

$$w_0^{(i)} = \frac{1}{N}, \quad i = 1, \ldots, N.$$

2. Importance Weighting

- At each time step k, we update the weights of particles based on the likelihood function. The likelihood is defined by the measurement model:

$$Y_k \mid X_k \sim U(GX_k, HX_k).$$

- If the observed value Y_k falls within the range $[GX_k, HX_k]$, the likelihood is proportional to:

$$w_k^{(i)} \propto \frac{1}{HX_k^{(i)} - GX_k^{(i)}}.$$

Otherwise, the weight is set to a small value (to prevent numerical instability).

- Normalizing the weights:

$$w_k^{(i)} = \frac{w_k^{(i)}}{\sum_{j=1}^{N} w_k^{(j)}}.$$

3. Resampling (if necessary)

- Compute the Effective Sample Size (ESS):

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} (w_k^{(i)})^2}.$$

- If $N_{\text{eff}} < \frac{N}{2}$, we perform multinomial resampling:
  - Sampling N particles with replacement from the current particle set with probability proportional to their weights.
  - Reseting all weights to $w_k^{(i)} = \frac{1}{N}$.

4. Propagation (State Transition)

- Apply the nonlinear population dynamics model:

$$X_{k+1}^{(i)} = R_k^{(i)} X_k^{(i)} (1 - X_k^{(i)}) + \text{process noise},$$

where $R_k^{(i)} \sim U(A, B)$ represents the random reproduction rate, with $A = 0.8$ and $B = 3.8$.

5. Estimation of X_k

- The estimated state is computed as the weighted mean:

$$\tau_k = \sum_{i=1}^{N} w_k^{(i)} X_k^{(i)}.$$

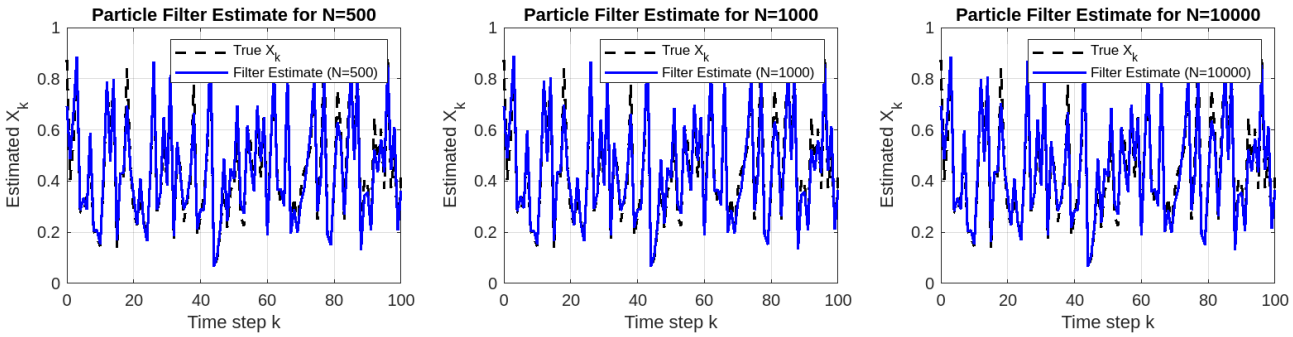
2.1.3 Results

The RMSE and MSE for each value of N are presented below:

N	RMSE	MSE
500	0.3332	0.2249
1000	0.3330	0.2244
10000	0.3347	0.2242

Additionally, the corresponding plots for each N are shown:

- $N = 500$
- $N = 1000$
- $N = 10000$



(a) $N = 500$ (b) $N = 1000$ (c) $N = 10000$

Figure 2: Comparison of filter estimates for different values of N.

The black dashed line represents the true state X_k, while the blue line represents the estimated values.

2.2 Conclusion

The results show that $N = 1000$ is a reasonable choice for this problem, as it provides a good balance between accuracy and computational cost. Increasing N beyond 1000 does not significantly reduce the RMSE, but it increases the computation time.

2.3 Part (b)

Control Variate

b) Use the technique on slide 25 of Lecture 7 to make a point wise confidence interval for X_k for $k = 0, 1, 2, \cdots, 100$. Check if the true X_k is between the upper and lower limit for $k = 0, 1, 2, \cdots, 100$. Try with N=500, 1000 and 10000. Which choice seems most reasonable?

2.3.1 Methodology

The particle filter is implemented to estimate the hidden state $\tau_k = E[X_k \mid Y_{0:k}]$ for time steps $k = 0, 1, \ldots, 100$. The filter follows these steps:

- **Initialization:** A set of ℓ particles is sampled uniformly.
- **Weight Update:** Particle weights are assigned based on the likelihood of the observed data.
- **Resampling:** If the effective sample size is below a threshold, particles are resampled to maintain diversity.
- **State Estimation:** The estimated state is computed as the weighted mean of the particles.
- **Confidence Interval Calculation:** Using the sorted weighted particles, a 95% confidence interval is computed for each time step.
- **Performance Evaluation:** Root Mean Square Error (RMSE) and the percentage of true values falling outside the confidence interval are computed.

2.3.2 Results and Discussion

The particle filter was tested for different values of ℓ, and the results are summarized below:

ℓ (Number of Particles)	MSE	Percentage Out of Bounds
500	0.2249	4.95%
1000	0.2244	3.96%
10000	0.2242	3.96%

Table 7: RMSE and Confidence Interval Performance for Different ℓ Values

2.4 Observations

- **MSE Analysis:** The MSE slightly decreases with increasing ℓ, indicating better estimation accuracy. However, the improvement from $\ell = 1000$ to $\ell = 10000$ is marginal.

- **Confidence Interval Reliability:** The percentage of true values falling outside the confidence interval is highest for $\ell = 500$ (4.95%) and improves for $\ell = 1000$ and $\ell = 10000$ (both at 3.96%).
- **Computational Consideration:** Increasing ℓ beyond 1000 provides only minor improvements while significantly increasing computational cost.

2.5 Conclusion

The best balance between accuracy and efficiency is achieved with $**\ell = 1000**$, which maintains a low MSE and a reliable confidence interval while keeping computational demands reasonable. Thus, $**\ell = 1000**$ is the recommended choice for implementing the particle filter in this problem.

References