Monte Carlo and Empirical Methods for Stochastic Inference

Home Assignment 2 Solution

Alessandro Celoria
Ruoyi Zhao

February 2023

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

1.1 Theoretical Concepts of SAWs

Upperbound of c_n We set out to demonstrate the following:

$$c_{n+m}(d) \leq c_n(d)c_m(d) \quad \forall n \geq 1,\, m \geq 1.$$

First, we consider $c_{n+m}(d)$ as an (n+m) steps self-avoiding walk of length $|S_{n+m}(d)|$ which is able to split into the self-avoiding path n that has already been taken and the following m steps that continue the self-avoiding walk from the n^{th} step. In this case, for the following steps m, we are adding the constraint, it can only continue walking from the n^{th} step onwards since the self-avoiding condition.
Conversely, $c_n(d)c_m(d)$ represents the n steps self-avoiding walks $|S_n(d)|$ and m steps self-avoiding walks $|S_m(d)|$, it means for all the possible combinations of the above two paths will include the possibility of paths overlapping. And based on the explanation above, this situation will not happen in $|S_{n+m}(d)|$.
Therefore, the possible walks satisfied when $n \geq 1$ and $m \geq 1$

$$|S_{n+m}(d)| \leq |S_n(d)||S_m(d)|$$

It implies that:

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

Limit of μ_d For every subadditive sequence $(a_n)_{n\geq 1}$, *Fekete's lemma* states that (let a_n be $c_n(d)$):

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

As previously discussed $c_{n+m}(d) \leq c_n(d)c_m(d)$, and by exploiting the continuity of the logarithm we can write:

$$log(\mu_d) = \lim_{n\to\infty} \frac{log(c_n(d))}{n}$$

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

Apparently, $\log(c_n(d))_{n\geq 1}$ is the subadditive sequence, so by using *Fekete's lemma* we are able to express our limit as:

$$log(\mu_d) = \lim_{n\to\infty} \frac{log(c_n(d))}{n} = \inf_{n\geq 1} \frac{log(c_n(d))}{n}$$

We then exploit the continuity of the exponential function to write $\mu_d = \lim_{n\to\infty} c_n(d)^{\frac{1}{n}}$.

General Bound of μ_d From the previous paragraph we remember:

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

In reference [3], it claims if n-step walks only take directions in the positive coordinate then it must be a self-avoiding walk. And for each step in dimension d, it is capable of choosing d destinations and we obtain d^n possible number of walks. Therefore, we have $d^n \geq c_n(d)$.
Furthermore, as a result of the set of n-step walks containing all possible paths, without immediate reversals, n-step self-avoiding walks, which has cardinality $(2d)(2d-1)^{n-1}$ ($2d$ directions for the first step and followed by $2d-1$ directions), it holds that $c_n(d) \leq (2d)(2d-1)^{n-1}$.
In conclusion of the above-mentioned, it comes to:

$$d^n \leq c_n(d) \leq (2d)(2d-1)^{n-1} \Rightarrow d \leq c_n(d)^{\frac{1}{n}} \leq (2d)^{\frac{1}{n}}(2d-1)^{\frac{n-1}{n}}$$

If we then take the limit for $n \to \infty$ we obtain:

$$\lim_{n\to\infty} d \leq \lim_{n\to\infty} c_n(d)^{\frac{1}{n}} \leq \lim_{n\to\infty} (2d)^{\frac{1}{n}}(2d-1)^{\frac{n-1}{n}} \Rightarrow d \leq \mu_d \leq 2d-1$$

General Bound of A_d For $d \geq 5$ we have:

$$c_n(d) = A_d \mu_d^n n^{\gamma_d - 1}$$

We take into account our previous finding that $c_{n+m}(d) \leq c_n(d)c_m(d)$ and therefore obtain:

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

Since it is proven that for $d \geq 5$ we have $\gamma_d = 1$ we can easily simplify the inequality as:

$$\begin{aligned} A_d \mu_d^{(n+m)}(n+m)^0 &\leq A_d \mu_d^n n^0 A_d \mu_d^m m^0 \\ A_d \mu_d^{(n+m)} &\leq A_d^2 \mu_d^{(n+m)} \\ A_d &\leq A_d^2 \Rightarrow 1 \leq |A_d| \end{aligned}$$

Since we know A_d is positive we have the following general bound:

$$A_d \geq 1, \text{ for } d \geq 5$$

1.2 Sequential Importance Sampling for SAWs

We consider a Monte Carlo approach to the estimation of the number of self-avoiding walks in $\mathbb{Z}^d$. Since we are talking about a sequence of steps in the considered vector space, it is natural to apply **Sequential Monte Carlo** methods to the problem. We look at the probability distribution that we have to deal with:

$$f_n(x_{0:n}) = \frac{z_n(x_{0:n})}{c_n} = \frac{\mathbb{1}_{S_n(d)}(x_{0:n})}{c_n(d)}$$

Of course we don't know the value of c_n as that's what we are trying to calculate. We therefore use Sequential Importance Sampling (SIS) which allows us to do exactly that. We then have to define a sequential instrumental PDF $g_n(x_{0:n})$. We can define two such functions: one trivial and one more efficient:

- **Trivial instrumental PDF** Pure random walk: choose the next move from a uniform distribution of all the $2d$ possible cardinal directions.
- **Trivial instrumental PDF** Locally self-avoiding walk: choose the next move from a uniform distribution of all the cardinal directions that do not lead to an already visited node.

Both these functions respect the requirements of an instrumental distribution, namely that it is nonzero whenever $z_n(x_{0:n})$ is nonzero. This can be easily verified as in the first case g_n would be the (normalized) indicator function for a valid random walk, which are a superset of SAWs, and in the second case it would be an indicator for locally-self-avoiding random walks, which holds the same property.

Target Function For the scope of this assignment we are only interested in calculating c_n, which is done through computation of the mean of the importance weights. We are not interested in any particular estimate that involves actually computing the expectation of an objective function, therefore we don't compute it nor define $\phi(X^{0:n})$ for this problem, rather we just update the weights and finally use them to estimate the normalizing constant c_n.

Weight updates To update the weights at the step $k+1$ we employ the following formulation:

$$w_{k+1}^i = w_k^i \frac{z_{k+1}(X_i^{0:k+1})}{z_k(X_i^{0:k}) g_{k+1}(X_i^{k+1}|X_i^{0:k})}$$

However we can greatly simplify the formula if we consider the meaning of $z_k(x)$:

- $w_k^i \neq 0 \Rightarrow z_k(X_i^{0:k}) = 1$ therefore we can simplify the denominator to remove dependence on z_k.
- $w_k^i = 0 \Rightarrow w_l^i = 0 \quad \forall l > k$, therefore if a weight drops to zero we can "kill" that particle.

Therefore our implementation follows the simplified logic shown in Algorithm 1. It is worth noting that since $g_n(X_i^{k+1}|X_i^{0:k})$ is a constant for the trivial instrumental PDF, the final weight is also going to be a constant equal to either 4^n for successful walks or 0 for unsuccessful walks. Since $c_n = \sum_{i=0}^{N} \left(w_i / \sum_{j=0}^{N} w_j \right)$ we can simplify this calculation to $c_n = N_A/N$, where N_A is the number of successful walks and N is the total number of walks attempted. This of course does not work for the smart g_n as the value of the distribution is not constant and changes with the number of "available" moves.

Algorithm 1 SIS algorithm (notice the change in notation of k)

$X^0 \sim g_o$
$w^0 = 0$

for $k = 0, 1, 2, \ldots$ **do**
  for $i = 1 \rightarrow N$ **do**
    if $w_{k-1}^i = 0$ **then**
      continue ▷ The updated weight would be zero anyways - no point in computing.
    end if

    $X_i^k \sim g_k(x_k|X_i^{0:k-1})$ ▷ Obtain the updated location

    if P_{new} *already visited* **then**
      $w_k^i \leftarrow 0$
    else
      $w_k^i \leftarrow \frac{w_{k-1}^i}{g_k(X_i^k|X_i^{0:k-1})}$
    end if
  end for
end for

With this code implemented and the two $g_n(x)$ defined we just need to run the simulation and then compute $c_n \sim \frac{1}{N} \sum_{i=1}^{N} w_n^i$. We performed the simulation with 10000 particles with sequences of up to 24 elements (which means a maximum of 23 steps) and repeated each experiment 10 times to get an idea of the variance.

Minimum	Average	Maximum
1	1	1
4	4	4
$1.1875 \cdot 10^1$	$1.2004 \cdot 10^1$	$1.2150 \cdot 10^1$
$3.5622 \cdot 10^1$	$3.6119 \cdot 10^1$	$3.6505 \cdot 10^1$
$9.8585 \cdot 10^1$	$1.0067 \cdot 10^2$	$1.0178 \cdot 10^2$
$2.7279 \cdot 10^2$	$2.8284 \cdot 10^2$	$2.9153 \cdot 10^2$
$7.5735 \cdot 10^2$	$7.8290 \cdot 10^2$	$8.1018 \cdot 10^2$
$2.0594 \cdot 10^3$	$2.1602 \cdot 10^3$	$2.2773 \cdot 10^3$
$5.6033 \cdot 10^3$	$5.8936 \cdot 10^3$	$6.1472 \cdot 10^3$
$1.4548 \cdot 10^4$	$1.5870 \cdot 10^4$	$1.7380 \cdot 10^4$
$3.9216 \cdot 10^4$	$4.4186 \cdot 10^4$	$4.6871 \cdot 10^4$
$1.1031 \cdot 10^5$	$1.2104 \cdot 10^5$	$1.3170 \cdot 10^5$
$3.0870 \cdot 10^5$	$3.2883 \cdot 10^5$	$3.5567 \cdot 10^5$
$7.3148 \cdot 10^5$	$8.6771 \cdot 10^5$	$1.0670 \cdot 10^6$
$2.1206 \cdot 10^6$	$2.2817 \cdot 10^6$	$2.4964 \cdot 10^6$
$5.5834 \cdot 10^6$	$6.3458 \cdot 10^6$	$7.3014 \cdot 10^6$
$1.0737 \cdot 10^7$	$1.6922 \cdot 10^7$	$2.1474 \cdot 10^7$
$4.1231 \cdot 10^7$	$5.0337 \cdot 10^7$	$5.8411 \cdot 10^7$
$8.2463 \cdot 10^7$	$1.1476 \cdot 10^8$	$1.5805 \cdot 10^8$
$2.7487 \cdot 10^8$	$3.6833 \cdot 10^8$	$5.2226 \cdot 10^8$
$3.2985 \cdot 10^8$	$8.1363 \cdot 10^8$	$1.5393 \cdot 10^9$
$1.7592 \cdot 10^9$	$2.3749 \cdot 10^9$	$3.5184 \cdot 10^9$
0	$6.8609 \cdot 10^9$	$1.4073 \cdot 10^{10}$
0	$1.4777 \cdot 10^{10}$	$2.8147 \cdot 10^{10}$

Steps
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24

Minimum	Average	Maximum
1	1	1
4	4	4
$1.2000 \cdot 10^1$	$1.2000 \cdot 10^1$	$1.2000 \cdot 10^1$
$3.6000 \cdot 10^1$	$3.6000 \cdot 10^1$	$3.6000 \cdot 10^1$
$9.9619 \cdot 10^1$	$9.9954 \cdot 10^1$	$1.0036 \cdot 10^2$
$2.8317 \cdot 10^2$	$2.8407 \cdot 10^2$	$2.8499 \cdot 10^2$
$7.7564 \cdot 10^2$	$7.8018 \cdot 10^2$	$7.8436 \cdot 10^2$
$2.1635 \cdot 10^3$	$2.1695 \cdot 10^3$	$2.1758 \cdot 10^3$
$5.8765 \cdot 10^3$	$5.9239 \cdot 10^3$	$5.9559 \cdot 10^3$
$1.6142 \cdot 10^4$	$1.6260 \cdot 10^4$	$1.6367 \cdot 10^4$
$4.3756 \cdot 10^4$	$4.4013 \cdot 10^4$	$4.4256 \cdot 10^4$
$1.1939 \cdot 10^5$	$1.2034 \cdot 10^5$	$1.2107 \cdot 10^5$
$3.2282 \cdot 10^5$	$3.2570 \cdot 10^5$	$3.2891 \cdot 10^5$
$8.7200 \cdot 10^5$	$8.7937 \cdot 10^5$	$8.9757 \cdot 10^5$
$2.3415 \cdot 10^6$	$2.3696 \cdot 10^6$	$2.3837 \cdot 10^6$
$6.3456 \cdot 10^6$	$6.4075 \cdot 10^6$	$6.4605 \cdot 10^6$
$1.7051 \cdot 10^7$	$1.7277 \cdot 10^7$	$1.7556 \cdot 10^7$
$4.5629 \cdot 10^7$	$4.6465 \cdot 10^7$	$4.7195 \cdot 10^7$
$1.2315 \cdot 10^8$	$1.2516 \cdot 10^8$	$1.2744 \cdot 10^8$
$3.3082 \cdot 10^8$	$3.3577 \cdot 10^8$	$3.4177 \cdot 10^8$
$8.8656 \cdot 10^8$	$8.9973 \cdot 10^8$	$9.1392 \cdot 10^8$
$2.3657 \cdot 10^9$	$2.4148 \cdot 10^9$	$2.4809 \cdot 10^9$
$6.2837 \cdot 10^9$	$6.4385 \cdot 10^9$	$6.5204 \cdot 10^9$
$1.6911 \cdot 10^{10}$	$1.7268 \cdot 10^{10}$	$1.7558 \cdot 10^{10}$

Table 1: SIS estimation of c_n. Left: with the trivial g_n. Right: with the locally self-avoiding g_n.

1.3 Introduction of Resampling

With the SISR (Sequential Importance Sampling and Resampling) technique we can avoid one of the systematic pitfalls of SIS: weights that degenerate due to them being obtained through a long chain of multiplications. To solve this, at each stage of the sequential calculation we re-sample with replacement from the current particle set using the weights as a probability distribution function. By doing that we promote particles that have higher weights (and are therefore more important) and use copies of those particles to replace other samples that instead are not very likely to occur. The downside of this is that we also increase the variance a little bit due to the stochastic resampling. The updated code to include resampling in the process is shown in Algorithm 2

1.4 Comparison

Figure 1 shows the high-level comparison between all three methods. For each method we plotted both the mean result as a solid line as well as two fictitious estimations as broken lines, comprised of the maximum and minimum estimation values for every point across all 10 runs, to build a sense of the variability of the estimations. Even from this very far away comparison we can see how towards the end the "confidence interval" of sorts for the SIS estimator with trivial g_n starts to fall apart. The reason for this is that by taking random walks most of them are going to crash as there is nothing there to prevent the walk from running into itself, and even if we completely ignore all steps except the previous one, there is still a 25% probability **at each step** of the walk backtracking and crashing. This is not a big problem for low counts of n, where a high particle number is going to more or less guarantee a number of successful runs sufficient to guarantee a stable approximation, but as we extend the length of the walk the anomalies become more common and the variance of the estimation increases radically. We can see this by looking at Table 2: for $n = 23$ and $n = 24$ with the trivial instrumental PDF the minimum value across all 10 runs is 0. With higher

Algorithm 2 SISR algorithm

$X^0 \sim g_o$
$w^0 = 0$

for $k = 0, 1, 2, \ldots$ **do**
  for $i = 1 \to N$ **do**
    $X_i^k \sim g_k(x_k | X_i^{0:k-1})$ ▷ Obtain the updated location

    if P_{new} *already visited* **then**
      $w_k^i \leftarrow 0$
    else
      $w_k^i \leftarrow \frac{1}{g_k(X_i^k | X_i^{0:k-1})}$
    end if
  end for

  Draw $\tilde{X}^{0:k}$ with replacement using the normalized weights as a discrete PDF

  $c_n \leftarrow c_n \cdot \frac{1}{N} \sum_{i=1}^{N} w_k^i$
  $w_k^i \leftarrow 1$ for $i = 1, 2, \ldots, N$
end for

Steps	Minimum	Average	Maximum
1	1	1	1
2	4	4	4
3	$1.2000 \cdot 10^1$	$1.2000 \cdot 10^1$	$1.2000 \cdot 10^1$
4	$3.6000 \cdot 10^1$	$3.6000 \cdot 10^1$	$3.6000 \cdot 10^1$
5	$9.9864 \cdot 10^1$	$1.0003 \cdot 10^2$	$1.0018 \cdot 10^2$
6	$2.8330 \cdot 10^2$	$2.8444 \cdot 10^2$	$2.8593 \cdot 10^2$
7	$7.7135 \cdot 10^2$	$7.8103 \cdot 10^2$	$7.8519 \cdot 10^2$
8	$2.1629 \cdot 10^3$	$2.1737 \cdot 10^3$	$2.1883 \cdot 10^3$
9	$5.8667 \cdot 10^3$	$5.9121 \cdot 10^3$	$5.9787 \cdot 10^3$
10	$1.6163 \cdot 10^4$	$1.6275 \cdot 10^4$	$1.6364 \cdot 10^4$
11	$4.3847 \cdot 10^4$	$4.4053 \cdot 10^4$	$4.4416 \cdot 10^4$
12	$1.1926 \cdot 10^5$	$1.2003 \cdot 10^5$	$1.2146 \cdot 10^5$
13	$3.2260 \cdot 10^5$	$3.2613 \cdot 10^5$	$3.3022 \cdot 10^5$
14	$8.6584 \cdot 10^5$	$8.8248 \cdot 10^5$	$9.0033 \cdot 10^5$
15	$2.3576 \cdot 10^6$	$2.3782 \cdot 10^6$	$2.4147 \cdot 10^6$
16	$6.3144 \cdot 10^6$	$6.4086 \cdot 10^6$	$6.5127 \cdot 10^6$
17	$1.7139 \cdot 10^7$	$1.7305 \cdot 10^7$	$1.7633 \cdot 10^7$
18	$4.5355 \cdot 10^7$	$4.6432 \cdot 10^7$	$4.7048 \cdot 10^7$
19	$1.2341 \cdot 10^8$	$1.2522 \cdot 10^8$	$1.2709 \cdot 10^8$
20	$3.2815 \cdot 10^8$	$3.3665 \cdot 10^8$	$3.4299 \cdot 10^8$
21	$8.8798 \cdot 10^8$	$9.0007 \cdot 10^8$	$9.2418 \cdot 10^8$
22	$2.3207 \cdot 10^9$	$2.4056 \cdot 10^9$	$2.4739 \cdot 10^9$
23	$6.3026 \cdot 10^9$	$6.4495 \cdot 10^9$	$6.6276 \cdot 10^9$
24	$1.6613 \cdot 10^{10}$	$1.7246 \cdot 10^{10}$	$1.7617 \cdot 10^{10}$

Table 2: Results of the SISR approach.

n values this becomes even more common as the estimator has to "get lucky" more times in a row in order to succeed. When we zoom in we can see the drastic difference: Lokking at the left plot of Figure 2 we can see that the confidence margin of the latter two methods is *not even visible* when we look at a meaningful

scale at the confidence level of the trivial g_n estimator's performance.

SISR Approach When we zoom in even further (right pane of Figure 2) we can see that SISR has a bit more of variance with respect to the "smart" SIS. This is expected as explained above, but other than that we don't see any significant difference with respect to SIS. The reason being that *in this particular problem* SIS does not incur in vanishing weights as we have that $z_k(x^{0:k}) = 1$ and $1/g(x_{k+1}|x_{0:k}) \geq 1$ at any given time a walk is active (that is, it has not crashed). Consequently, weight updates can only either increase the weight or set it to zero and thus applying SISR (which is an improvement specifically aimed at solving the vanishing weights problem) does not yield any appreciable improvement.

Overall Trend Taking a step back we can see how the logarithm of c_n seems to have a somewhat linear dependency on n if we ignore the first few values of n, and this appears to be confirmed by all three estimations methods which generally agree with each other pretty well.

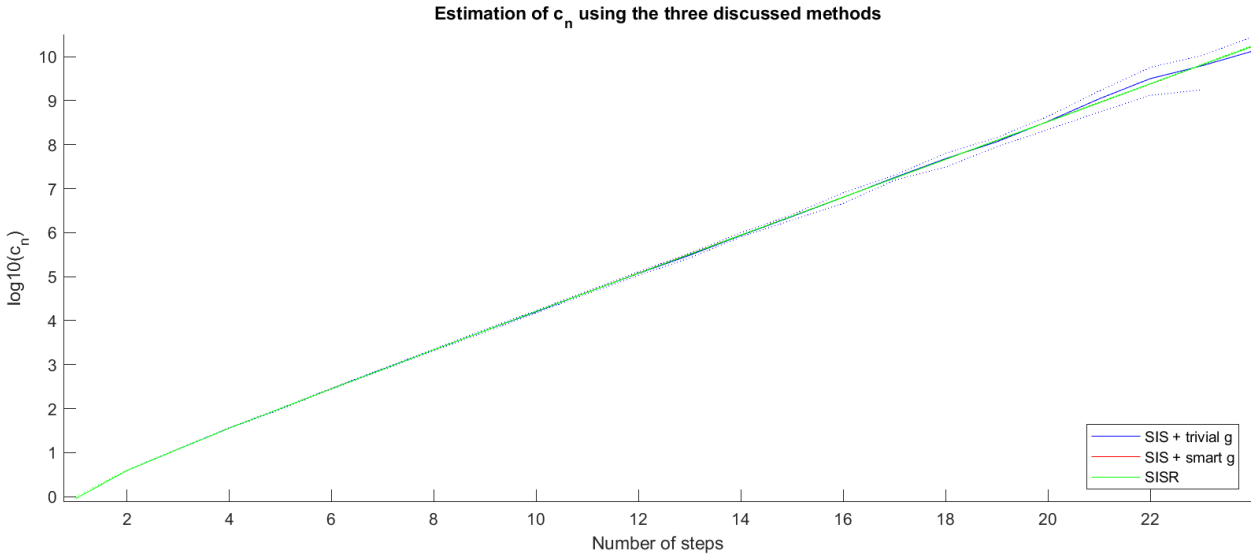

Figure 1: High level comparison of all three estimators. Notice the y axis is a log value.

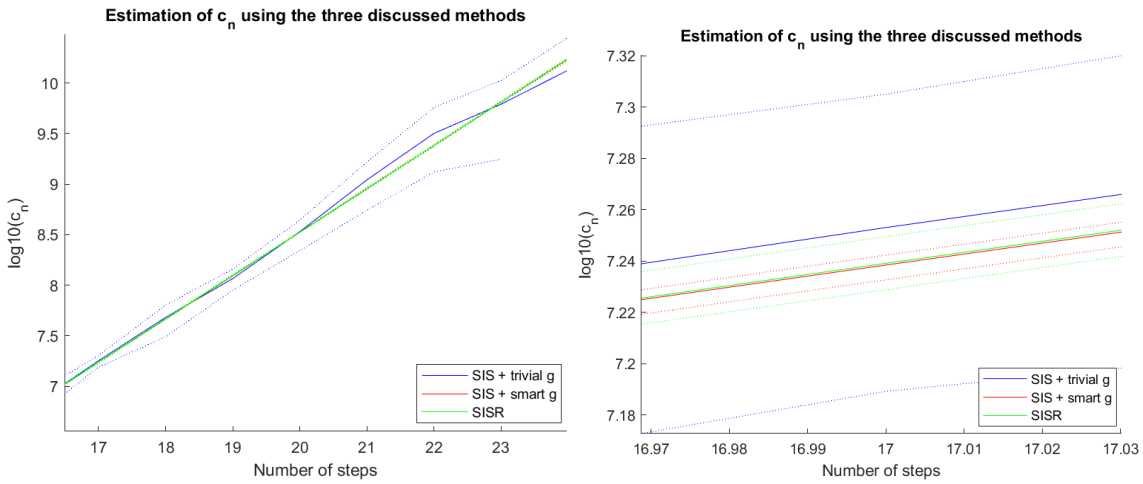

Figure 2: Close up comparison. Left: at the end of the run. Right: middle of the run, focus on SIS with locally self-avoiding g and SISR with the same instrumental PDF.

1.5 Parameter Estimation

We take a look at the formula for c_n:

$$c_n \approx A_d \mu_d^n n^{\gamma_d - 1} \quad \forall d \in \mathbb{N},\ d \neq 4$$
$$ln(c_n) \approx ln(A_d) + n \cdot ln(\mu_d) + ln(n) \cdot (\gamma_d - 1)$$

Our goal is now to estimate the values of A_d, μ_d and γ_d. If we instead consider $ln(A_d)$, $ln(\mu_d)$ and $(\gamma_d - 1)$ as well as $ln(c_n)$ and choose as inputs 1, n and $ln(n)$ we can cast the problem as a linear regression:

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 2 & 0.69 \\ & \vdots & \\ 1 & n & ln(n) \end{bmatrix} \cdot \begin{bmatrix} ln(A_d) \\ ln(\mu_d) \\ \gamma_d - 1 \end{bmatrix} = \begin{bmatrix} ln(c_1) \\ ln(c_2) \\ \vdots \\ ln(c_n) \end{bmatrix}$$

As before we compute c_n 10 times for each n, then we calculate these parameters 10 times independently and perform an average to find our final estimates for A_d, μ_d and γ_d:

$d=2$	A_2	μ_2	γ_2
SIS	1.48	2.655	1.2327
$SISR$	1.4795	2.6546	1.234

$d=2$	A_2	μ_2	γ_2
SIS	$5.1917 \cdot 10^{-6}$	$3.3805 \cdot 10^{-6}$	$1.3627 \cdot 10^{-5}$
$SISR$	$5.1531 \cdot 10^{-6}$	$5.8656 \cdot 10^{-6}$	$2.544 \cdot 10^{-5}$

$d=3$	A_3	μ_3	γ_3
$SISR$	1.2635	4.6952	1.1262

$d=3$	A_3	μ_3	γ_3
$SISR$	$3.2348 \cdot 10^{-6}$	$2.0277 \cdot 10^{-6}$	$6.6897 \cdot 10^{-6}$

Table 3: On the left: mean values of the parameters as estimated for both $d = 2$ and $d = 3$. On the right, the corresponding variance of each estimate.

Looking at the variance we can see how γ_n seems to be consistently harder to estimate than the other two, although with $d = 3$ the gap with the variance of the other two parameters closes up a bit. The values of μ_d found here respect the bounds previously discussed.

2 Filter Estimation

A Hidden Markov Model comprises two stochastic processes:

- A Markov Chain $(X_k)_{k\geq 0}$ with transition density q, where X_k is the relative population size at generation k. Due to the Markov property, X_k conditioned on X_{k-1} is independent from any previous state or any past or current sensor output.
- An Observer Model $(Y_k)_{k\geq 0}$ where, due to the Markov property, Y_k conditioned on X_k is independent from any previous hidden state or sensor measurement.

We use the following models and initial distribution, that are given to us:

- **Transition Model:** $X_{k+1} = B_{k+1} X_k (1 - X_k)$ with $B_{k+1} \in U(0.9, 3.9)$ iid and $k = 0, 1, 2, \ldots$
- **Observer Model:** $Y_k | X_k = y \sim U(0.7x, 1.2x)$
- **Initial State:** $X_0 \sim U(0.6, 0.99)$

We apply the SISR algorithm to the Y_k and X_k sequences we got from the provided `population_2023.mat` file. To compute the expectation of the population given a sequence of readings we use the following formula:

$$\mathbb{E}\{X_k|Y_{0:k}\} = \tau_k = \sum_{i=1}^{N} \frac{\omega_k^i}{\sum_{t=1}^{N} \omega_k^l} \phi(X_i^{0:k})$$

The weights are computed through an *observation density* $p(y_k|x_k)$ which in our case is the observer model. By adapting the code segments that have been suggested to us for the completion of this task for $n = 1 \ldots 50$ we obtained the results show in Table 4. This data includes a 95% confidence interval that we calculated with the following code taken from *Page 25 of the L7 Slide Deck*:

```
[xx,I]=sort(part); % sort data
cw=cumsum(w(I))/sum(w); % cumulative normalized weight sum for sorted data

Ilower=find(cw>=0.025,1); % index for the lower 2.5% quantile
Iupper=find(cw>=0.975,1); % index for the upper 2.5% quantile
taulower(k+1)=xx(Ilower); % lower 2.5% quantile
tauupper(k+1)=xx(Iupper); % upper 2.5% quantile
```

As we can see the pairs $(X_i^{0:k}, \omega_k^i)_{i=1}^N$ not only play a crucial role for resampling, but also for the computation of the confidence interval.

Results In Figure 3 we can see how most of the values of the true X_k sequence (blue line) are in the confidence interval (pink lines), except for $k = 10$, $k = 43$, so we can say that this approximated 95% confidence interval is a good construction.

k	0	1	2	3	4	5	6	7	8	9
LB	0.6099	0.5416	0.5305	0.4374	0.2515	0.1762	0.3791	0.3674	0.6840	0.2352
X_k	0.7259	0.5625	0.5373	0.4627	0.2969	0.1933	0.4245	0.3880	0.8501	0.3896
UB	0.9667	0.8647	0.8783	0.7287	0.4180	0.2681	0.6259	0.6131	0.9566	0.3873
k	10	11	12	13	14	15	16	17	18	19
LB	0.4781	0.6398	0.1152	0.4566	0.3620	0.2183	0.5896	0.4892	0.3250	0.2288
X_k	0.4878	0.8365	0.1837	0.4620	0.5874	0.2615	0.5941	0.5201	0.3310	0.3167
UB	0.7953	0.9556	0.1926	0.5923	0.5978	0.3052	0.7975	0.8145	0.5436	0.3867
k	20	21	22	23	24	25	26	27	28	29
LB	0.3491	0.4105	0.7271	0.4304	0.2185	0.2137	0.4523	0.4480	0.1837	0.5089
X_k	0.3832	0.4851	0.7347	0.6705	0.2890	0.3192	0.5888	0.5176	0.2630	0.6603
UB	0.4105	0.6892	0.9575	0.6744	0.3422	0.3578	0.7513	0.7472	0.2658	0.7239
k	30	31	32	33	34	35	36	37	38	39
LB	0.5515	0.5412	0.3147	0.5222	0.2172	0.2407	0.2364	0.2015	0.1706	0.1582
X_k	0.6561	0.5738	0.3192	0.7944	0.2566	0.3807	0.2494	0.2319	0.2401	0.2520
UB	0.9176	0.8908	0.5146	0.8746	0.3637	0.3962	0.3955	0.3380	0.2790	0.2552
k	40	41	42	43	44	45	46	47	48	49
LB	0.5030	0.2668	0.2032	0.4567	0.3680	0.8687	0.0515	0.1066	0.4752	0.2256
X_k	0.6155	0.2887	0.2017	0.5170	0.5813	0.9365	0.0583	0.1536	0.4942	0.2923
UB	0.7133	0.4362	0.3397	0.7428	0.6060	0.9580	0.0862	0.1786	0.5561	0.2944

Table 4: Filter Expectation τ_k for k = 0,1,2,...,49

Then the estimation of the filter expectation and the 95% confidence interval can be observed in Figure 1.

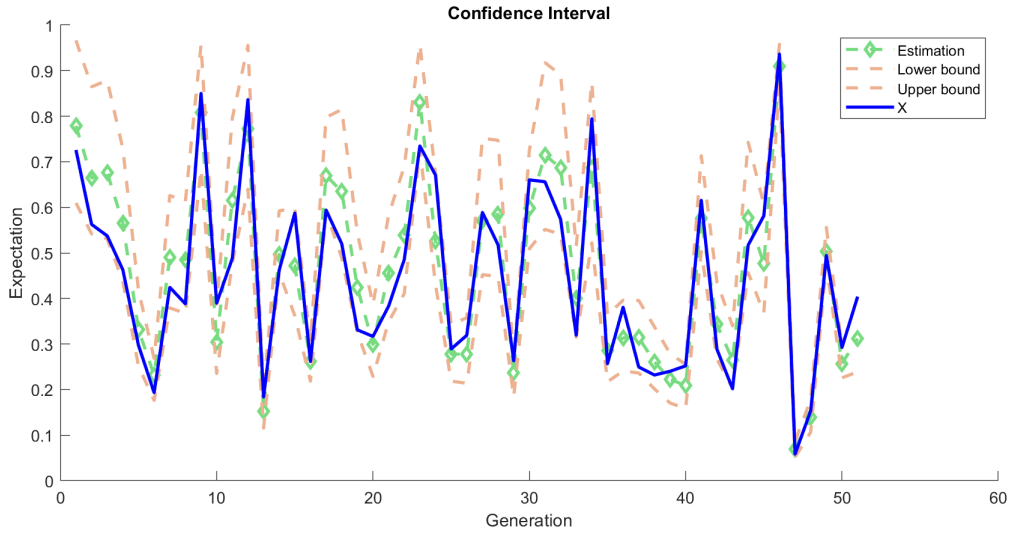

Figure 3: Comparison of the confidence interval for τ_k at generation k with true value X_k.