Homework Assignment 2

Alfred Bornefalk
Simen Söderberg

21 February 2022

Contents

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

1.1 Problem 1: The Submultiplicative Property

In this part we aim to convince ourselves that, for all $n \geq 1$ and $m \geq 1$, the following holds:

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

It is given that $|S_n(d)| = c_n(d)$. Thus, $|S_n(d)||S_m(d)| = c_n(d)c_m(d)$ for possible walks in $\mathbb{Z}^d$. To show the property in (1), after completing n steps, we let the m following steps be conditioned on the n steps taken in the previous walk, where the first of these steps originate from the position of the n^{th} step. Because of the added criteria to the walk, we can directly observe that we have at least one less possible step that we can take, since we are not allowed to go back to the point of the $n-1^{\text{st}}$ step. As a direct result of this, we can realize that the modified possible walks $|S_m^*(d)|$ satisfy

$$|S_m^*(d)| \leq |S_m(d)| \tag{2}$$

for all $n \geq 1$ and $m \geq 1$. Of course, we could perform the argument the other way around; first walking m steps, and then letting the following n steps be conditioned on the previous walk. Thus, by using (2), we obtained the desired result in (1) through

$$c_{n+m}(d) = |S_n(d)||S_m^*(d)| \leq |S_n(d)||S_m(d)| = c_n(d)c_m(d)$$

The authors are hereby convinced that the given property holds.

1.2 Problem 2: Using Fekete's Lemma

A subadditive sequence $(a_n)_{n\geq 1}$ satisfies

$$a_{n+m} \leq a_n + a_m \tag{3}$$

Fekete's lemma states that for every subadditive sequence $(a_n)_{n\geq 1}$ the limit

$$\kappa_d = \lim_{n\to\infty} \frac{a_n}{n} \tag{4}$$

exists. Taking the logarithms of (1) gives

$$\log c_{n+m}(d) \leq \log c_n(d) + \log c_m(d) \tag{5}$$

Clearly, comparing (5) to (3), $(\log c_n(d))_{n\geq 1}$ is as a subadditive sequence. Applying this to (4), the expression for our limit is

$$\kappa_d = \lim_{n\to\infty} \frac{1}{n} \log c_n(d) = \lim_{n\to\infty} \log c_n(d)^{\frac{1}{n}} \tag{6}$$

Since μ_d can be derived from κ_d according to

$$\log_b \mu_d = \kappa_d \iff \mu_d = b^{\kappa_d}$$

for an arbitrary base b, the expression for μ_d is obtained through (6) by

$$\mu_d = \lim_{n\to\infty} b^{\log_b c_n(d)^{\frac{1}{n}}} = \lim_{n\to\infty} c_n(d)^{\frac{1}{n}} \tag{7}$$

Thus, by using Fekete's lemma, we have proved that the limit in (7) exists.

1.3 Problem 3: Sequential Importance Sampling

The constant μ_d is called the connectivity constant, and it depends on the particular lattice chosen. In the home assignment, we are investigating the lattice $\mathbb{Z}^d$. For $d = 2$, as $n \to \infty$, it is conjectured that

$$c_n(2) \to A_2 \mu_2^n n^{\gamma_2 - 1} \tag{8}$$

We are now asked to estimate the $c_n(2)$'s with the sequential important sampling algorithm (SIS). We achieve this by letting the instrumental distribution g_n be 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 neighbors of X_k. Since there is no self-avoiding mechanism in the random walk, the method estimate each $c_2(n)$ by first simply simulating a large number N of random walks in $\mathbb{Z}^2$. Then, letting N_{SA} denote the number of self-avoiding ones, the ratio of self-avoiding walks amongst the random walks for every $n = 1, 2, 3, ...$ is given by $\frac{N_{SA}}{N}$. Since we, by definition of performing a random walk in $\mathbb{Z}^2$, always have 4 possible destinations for the step $k + 1$, $c_n(2)$ can now be estimated as

$$c_n(2) = 4^n \frac{N_{SA}}{N} \tag{9}$$

for each n. An extraction of the estimates of $c_n(2)$'s, up until $n = 10$, when applying the described SIS technique and (9) in MATLAB, using $N = 10^4$, is presented in table 1. These are compared to the true values of c_n on the 2-dimensional square lattice presented by Slade (1994).

Table 1: The number of estimated possible walks $c_n(2)$ when taking n steps with the sequential importance sampling algorithm as well as the true theoretical value of self-avoiding walks, for $n = \{1, 2, ..., 10\}$.

Steps	Possible Walks	True Value
1	4.00	4
2	11.87	12
3	36.40	36
4	99.17	100
5	291.23	284
6	783.97	780
7	2,179.07	2,172
8	5,950.67	5,916
9	15,833.50	16,268
10	42,991.62	44,100

From the values presented in table 1 of $c_n(2)$ up to $n = 10$ steps, the reliability of the described SIS algorithm can be questioned, since the estimates of possible walks clearly deviate from its true value. We can further confirm that this is indeed a naive approach by using (8), rewriting the formula as

$$\frac{c_{n+1}(2)}{c_n(2)} \to \frac{A_2\mu_2^{n+1}(n+1)^{\gamma_2-1}}{A_2\mu_2^n n^{\gamma_2-1}} = \mu_2(\frac{n+1}{n})^{\gamma_2-1}$$

Solving for μ_2 and using the known fact that $\gamma_2 = \frac{43}{32}$, we get

$$\mu_2 \to \frac{c_{n+1}(2)}{c_n(2)}(\frac{n}{n+1})^{\frac{11}{32}} \tag{10}$$

When increasing n, we can see that the deviations from the mean estimate of μ_2 should get smaller and smaller. Thus, we implement (10) up to $n = 25$ in MATLAB. The corresponding plot is showcased in figure 1.

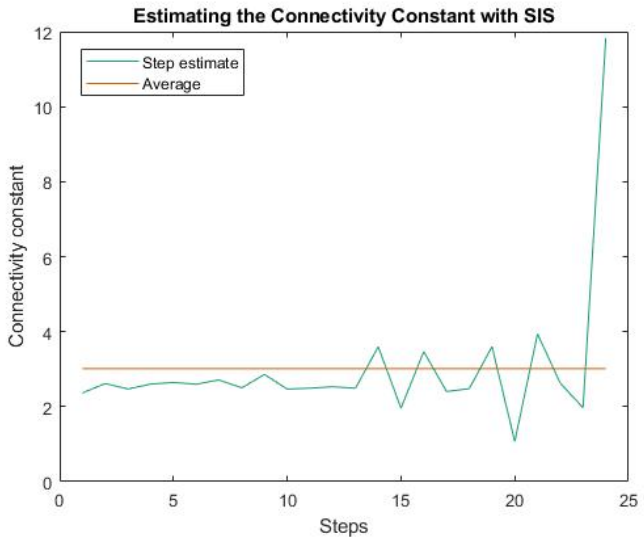

Figure 1: 2D plot of μ_2 estimated from the SIS algorithm as n goes from 1 to 25. The green graph shows the estimate of the connectivity constant for each step n, whilst the orange horizontal line is the average of the interval.

By observing figure 1, we can see that μ_2 seems to be somewhat consistent for the interval presented in table 1. However, from this point, the estimates start to deviate more and more from the mean. Because of the fact that the amplitudes keep increasing when n increases, our conclusion is that the SIS algorithm is not a great method for estimating the connectivity constant. In fact, when estimating with $n > 25$, we rarely find a self-avoiding walk at all.

1.4 Problem 4: Self-Avoiding Random Walk

Now, we are to improve the naive approach in the previous subsection. We do so by letting the g_n in our SIS algorithm be the distribution of a self-avoiding random walk $(X_k)_{k=0}^n$ in $\mathbb{Z}^2$. The implication of this modification is (i) $X_0 = 0$, and (ii) given $X_{0:k}$, the next step X_{k+1} is drawn uniformly among the set of free neighbours $\mathbf{N}(X_{0:k})$. If $\mathbf{N}(X_{0:k}) = \emptyset$, then X_{k+1} is simply set to X_k. With our improved SIS algorithm, the SIS updates the estimator

$$\sum_{i=1}^{N} \frac{\omega_n^i}{\sum_{l=1}^{N} \omega_n^l} \phi(X_i^{0:n}) \approx \mathbb{E}_{f_n}(\phi(X_{0:n}))$$

to the estimator

$$\sum_{i=1}^{N} \frac{\omega_{n+1}^i}{\sum_{l=1}^{N} \omega_{n+1}^l} \phi(X_i^{0:n+1}) \approx \mathbb{E}_{f_{n+1}}(\phi(X_{0:n+1}))$$

by only adding a component X_i^{n+1} to $X_i^{0:n}$ and sequentially updating the weights. It is to be noted that for each n an estimate of c_n can be obtained as

$$\frac{1}{N} \sum_{i=1}^{n} \omega_n^i \approx c_n \tag{11}$$

Consequently, if we are able to derive the importance weighting functions w_n's, we can solve for the c_n's according to (11). The updated weights are in the general case obtained by computing

$$\omega_{n+1}^i = \frac{z_{n+1}(X_i^{0:n+1})}{z_n(X_i^{0:n}) g_{n+1}(X_i^{n+1} | X_i^{0:n})} \omega_n^i \tag{12}$$

where the initial weight, w_1, is set to 1. In our case, we let z_k be 1 if there are any available neighbors for step k, and 0 otherwise. Letting A denote the number of available neighbors, g_k is given by

$$g_k = \frac{1}{A}, A \geq 1 \tag{13}$$

In the above equation, we note that when there are no available neighbours, i.e. $A = 0$, the updated weights ω_{k+1}^i up until $k = n$ are set to zero. This comes directly from the fact that $A = 0 \implies z_{k+1} = 0$. Thus, there is no need to define g_k for $A = 0$. Implementing MATLAB code that keeps track of the available neighbors in the self-avoiding random walk yields values for the binary variables z_k's, as well as the instrumental densities, defined by (13). Thus, we are able to compute weights as described in (12), and finally provide estimates of the number of possible walks for different steps. These results, again compared with the true values (Slade), are displayed in table 2 up to $n = 10$. In order to maintain computational efficiency, we set $N = 10^3$.

Table 2: The number of possible walks $c_n(2)$ when taking n steps with the improved sequential importance sampling algorithm as well as the true theoretical value of self-avoiding walks, for $n = \{1, 2, ..., 10\}$.

Steps	Possible Walks	True value
1	4.00	4
2	12.00	12
3	36.00	36
4	99.88	100
5	283.72	284
6	779.13	780
7	2,167.92	2,172
8	5,879.08	5,916
9	16,189.47	16,268
10	43,952.77	44,100

Observing table 2, we can directly see that our algorithm correctly estimates $c_1(2) = 4$, $c_2(2) = 12$ and $c_3(2) = 36$. Furthermore, it is not far away from estimating the true theoretical values of $c_4(2) = 100$ and $c_5(2) = 284$. Of course, this analysis is not rigorous enough to draw a conclusion on whether the improved SIS algorithm is efficient enough or not. Thus, we will again examine the estimated connectivity constant. This time around, the method keeps providing non-zero estimates for $n > 25$. Thus, a MATLAB figure for steps up to $n = 50$ is presented below.

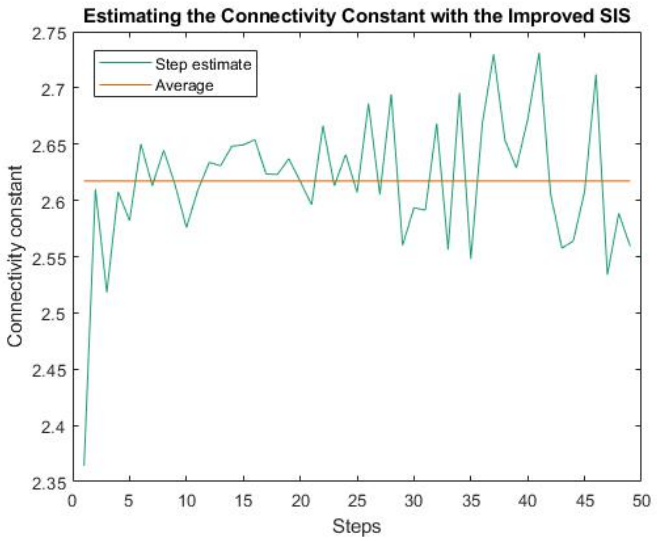

Figure 2: 2D plot of μ_2 estimated from the improved SIS algorithm as n goes from 1 to 50. The green graph shows the estimate of the connectivity constant for each n, whilst the orange horizontal line is the average of the interval.

Comparing figure 2 to figure 1, it is evident that the improved SIS algorithm, where we have considered a self-avoiding random walk, yields better estimates than those presented earlier. However, as n increases, the deviations from the mean increases. Thus, it seems to be a rather good estimator for small values of n. Increasing n even further, assuming that the computer is strong enough to perform the computations in an acceptable amount of time, it is deemed highly likely that the deviations from the mean will continue to increase. We conclude that this approach yields estimates with much greater precision than the previous algorithm. As we are about to see however, we are not quite yet done with the optimization of our SIS algorithm.

1.5 Problem 5: Sequential Importance Sampling With Resampling

In problem 5 we aim to expand our previous sequential importance sampling algorithm with resampling. To implement our resampling algorithm, we first draw with replacement new particles $\tilde{X}_1^{0:n}, \tilde{X}_2^{0:n}, ..., \tilde{X}_N^{0:n}$ from the particles obtained through sequential importance sampling $X_1^{0:n}, X_2^{0:n}, ..., X_N^{0:n}$ with the probabilities from our normalized importance weights, defined as $\frac{\omega_n^j}{\sum_{l=1}^N \omega_n^l}$. Thus, for each new particle, the probability that it is equal to one of the previously obtained particles is

$$\mathbb{P}[\tilde{X}_i^{0:n} = X_j^{0:n}] = \frac{\omega_n^j}{\sum_{l=1}^N \omega_n^l}$$

The second step is to draw, for all i, $X_i^{0:n+1}$ from our instrumental distribution conditioned on our resampled particles $X_i^{0:n+1} \sim g_{n+1}(x_{n+1}|\tilde{X}_i^{0:n})$. Thereafter, we set, for all i, $X_i^{0:n+1} = (\tilde{X}_i^{0:n}, X_i^{n+1})$ and update our weighting function similarly as for the sequential importance sampling

$$\omega_{n+1}^i = \frac{z_{n+1}(X_i^{0:n+1})}{z_n(\tilde{X}_i^{0:n}) g_{n+1}(X_i^{n+1}|\tilde{X}_i^{0:n})} \tag{14}$$

Again, when there are no possible self-avoiding steps, the updated weighting function ω_{k+1}^i up until $k = n$ is set to zero. This sampling technique yields an estimate of the number of possible walks in the 2-dimensional square lattice; estimates up to $n = 10$ are presented in table 3. We are again using $N = 10^3$, and the results are compared with the true possible walks (Slade).

Table 3: The number of possible walks $c_n(2)$ when taking n steps with the sequential importance sampling algorithm with resampling as well as the true theoretical value of self-avoiding walks, for $n = \{1, 2, ..., 10\}$.

Steps	Possible Walks	True Value
1	4.00	4
2	12.00	12
3	36.00	36
4	99.91	100
5	284.35	284
6	780.22	780
7	2,168.14	2,172
8	5,883.91	5,916
9	16,147.80	16,268
10	43,594.22	44,100

From table 3, we can once again see that our estimates for $n = \{1, 2, 3\}$ are extremely accurate. The results obtained by the resampling algorithm does not seem to be much better compared to the values in table 2 from our improved SIS algorithm for $n = \{1, 2, ..., 10\}$. However, an important remark here is that we only simulate for small values of n. As we will see when we compare the connectivity constant for the improved SIS and our SISR algorithm with larger n's, the discrepancies between the models will be more distinct. Once again we examine the estimated connectivity constant; the MATLAB figure up to $n = 50$ is presented below.

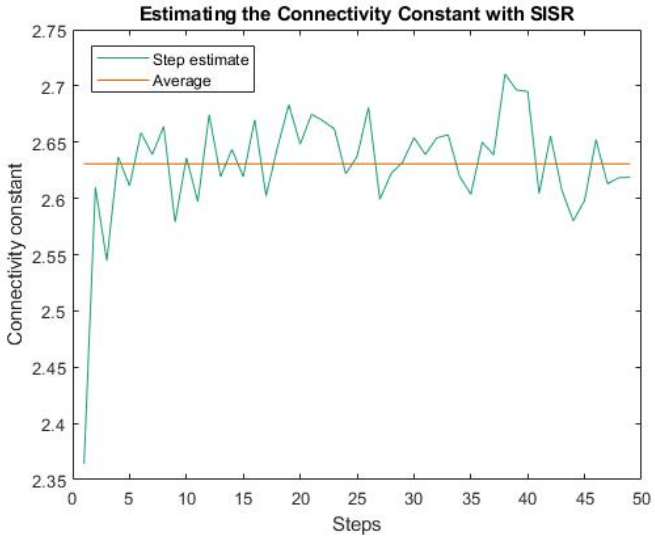

Figure 3: 2D plot of μ_2 estimated from the SISR algorithm as n goes from 1 to 50. The green graph shows the estimate of the connectivity constant for each n, whilst the orange horizontal line is the average of the interval.

By comparing figure 2 and 3 it is hard to draw any conclusions if the estimation of the connectivity constant has improved or not with the resampling method. Here, one way to evalutate the difference between the accuracy of the methods is to increase the maximum number of steps n. We do so by estimating the connectivity constant for the improved SIS algorithm and the SISR algorithm for $n = \{1, 2, ..., 100\}$. The resulting plot is presented below in figure 4.

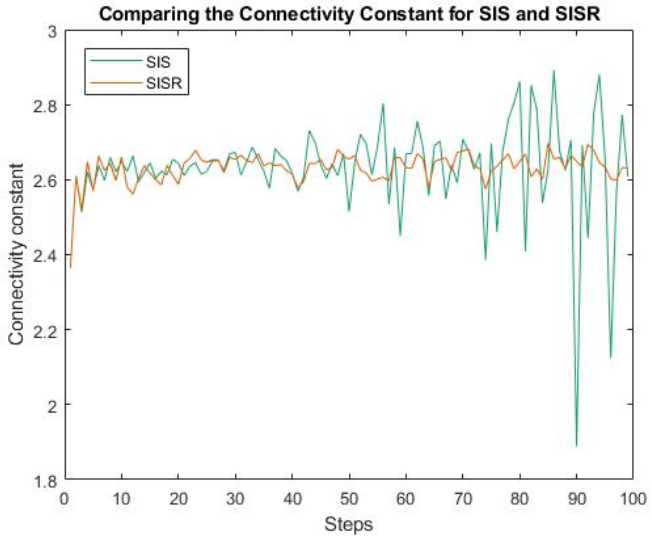

Figure 4: 2D plot of μ_2 estimated from the improved SIS algorithm and the SISR algorithm as n goes from 1 to 100. The green graph shows the estimate of the connectivity constant with the improved SIS algorithm for each n, whilst the orange graph shows the estimate of the connectivity constant with the SISR algorithm for each n.

Observing figure 4, we can see that the estimated connectivity constant initially is comparable for the improved SIS algorithm and the SISR algorithm. For larger n's however, the SIS algorithm yields more and more deviating estimates, one time even estimating μ_2 to be less than 2. The SISR algorithm on the other hand provides considerably less volatile μ_2 estimates. This suggests that the distribution of the importance weights originating from the improved SIS algorithm has larger tails, which is in line with the theory presented by Wiktorsson (2022, 16). As a consequence of the wider distribution, the probability of getting inaccuracies from the true value in the importance weighting function increases, leading to more and more noticeable offsets in the μ_2 predictions. Finally, we conclude that sequential importance sampling with resampling outperforms the sequential importance sampling algorithm when estimating the possible walks in the 2-dimensional square lattice, especially for larger n's.

1.6 Problem 6: Obtaining Estimates

We are now supposed to use our SISR estimates of the $c_n(2)$'s from the above subsection in order to obtain estimates of A_2, μ_2, and γ_2. First, we transform the parameters by taking the logarithms of (8), yielding

$$\log(c_n(2)) = \log(A_2) + n\log(\mu_2) + (\gamma_2 - 1)\log(n) \tag{15}$$

Now, we aim to identify a linear regression on the form

$$y = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 \tag{16}$$

For our estimation, we are using $n = \{1, 2, ..., 100\}$. Using (15), our independent and dependent variables are thereafter, $\forall n$, set to

$$y = \log(c_n(2)), x_1 = 1, x_2 = n, x_3 = \log(n) \tag{17}$$

By using the built-in multiple linear regression method in MATLAB, we receive the following coefficient estimates:

$$\beta_1 = .3340, \beta_2 = .9720, \beta_3 = .2831$$

Plugging these in to (16) combined with our expressions in (17), we get the linear regression model

$$y = .3340 + .9720n + .2831\log(n) \tag{18}$$

Finally, A_2, μ_2, and γ_2 can be derived through (15) and (18). For A_2, we get

$$\log(A_2) = .3340 \iff A_2 = e^{.3340} \approx 1.40$$

The connective constant μ_2 is given by

$$n\log(\mu_2) = .9720n \iff \mu_2 = e^{.9720} \approx 2.64$$

Lastly, γ_2 is computed according to

$$(\gamma_2 - 1)\log(n) = .2831\log(n) \iff \gamma_2 = 1 + .2831 \approx 1.28$$

Regarding which of the three parameters that is the easiest one to estimate, we recall equation (8):

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

From this, we can see that A_2 grows linearly, whilst μ_2 behaves like a power function and γ_2 like an exponential function. It is common knowledge that power functions and exponential functions grow faster than linear functions. Furthermore, since we have very small values of the independent variables, we expect the power function to grow faster than the exponential function in our interval. The implications of these arguments are that μ_2 should be the easiest parameter to estimate, whereas A_2 ought to be the hardest one. This is because,

holding A_2 and γ_2 fixed, small changes in the value of μ_2 should (asymptotically) satisfy (8) when $c_n(2)$ is updated. Vice versa, holding μ_2 and γ_2 constant, A_2 needs to change more in order to compensate for updates of $c_n(2)$. In order to test our hypothesis, we perform 10 independent simulations, repeating the procedure outlined above. The minimum and maximum parameter estimates, as well as the sample variance, are presented in table 4.

Table 4: The lowest and highest observed value as well as the variance over 10 simulations of the parameter estimates A_2, μ_2, and γ_2. The underlying $c_n(2)$'s are computed with the SISR algorithm with $N = 10^3$ and $n = \{1, 2, ..., 100\}$.

Parameter	Lowest	Highest	Variance
A_2	1.3188	1.5024	.0140
μ_2	2.6335	2.6486	.0001
γ_2	1.2514	1.3318	.0048

Looking at table 4, our hypothesis that μ_2 should be the easiest parameter to estimate is confirmed. The variance of the μ_2 estimates is considerably lower than for the other two parameters. Additionally, further supporting the previous analysis, there is a strong indication that A_2 is in fact the hardest parameter to obtain reliable estimates for, since its variance is three times as large as the variance of γ_2. As is usually the case, the variance for all parameters could of course be reduced through increasing N. As $N \to \infty$, the estimates of the $c_2(n)$'s will be more and more consistent, leading to less variation in our parameter estimates. Naturally, this is not practically possible, and in our case not even needed. The current choice of N supports the aforementioned hypothesis, and still provides relatively stable estimates for A_2 and γ_2. (We do know from the instructions in the home assignment that $\gamma_2 = \frac{43}{32}$, and our estimates of γ_2 are considered acceptably close to this value.)

1.7 Problem 7: Verifying a General Bound

We aim to verify that the following general bound should hold:

$$d \leq \mu_d \leq 2d - 1 \tag{19}$$

We begin by imagining ourselves a walk where we can only head in a positive direction. That is, for any dimension d, we can for each step choose d destinations. Thus, for a walk of length n, the possible amount of walks we can take is d^n. Clearly, since our walk is limited, it must hold that $d^n \leq c_n(d)$. Now, we imagine a walk where we are allowed to walk in both a positive and negative direction, but we cannot go back to our previous step. Here, we can choose $2d$ directions for our first step, and thereafter $2d - 1$ directions, making the possible amount of walks equal to $2d(2d-1)^{n-1}$. Since we in this case only apply the limitation that we cannot go back to our previous step, it is evident that $c_n(d) \leq 2d(2d-1)^{n-1}$. With these arguments, we have obtained the inequality

$$d^n \leq c_n(d) \leq 2d(2d-1)^{n-1} \iff d \leq c_n(d)^{\frac{1}{n}} \leq (2d(2d-1)^{n-1})^{\frac{1}{n}} \tag{20}$$

In order to verify (19), we examine how (20) behaves when $n \to \infty$. For $c_n(d)^{\frac{1}{N}}$, the result is obtained immediately from (7):

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

The limit for $(2d(2d-1)^{n-1})^{\frac{1}{n}}$ is obtained through basic calculus:

$$\lim_{n\to\infty} (2d(2d-1)^{n-1})^{\frac{1}{n}} = \lim_{n\to\infty} (\frac{2d}{2d-1})^{\frac{1}{n}}(2d-1) = (2d-1)$$

Plugging our asymptotic values into (20), we get

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

Thus, we have shown that the general bound displayed in (19) should hold.

1.8 Problem 8: Verifying Another General Bound

In problem 8 we aim to show that the following general bound should hold

$$A_d \geq 1, \text{for } d \geq 5 \tag{21}$$

To verify this, we leverage the fact that the following is true as $n \to \infty$:

$$c_n(d) \sim \begin{cases} \mathrm{A}_d \mu_d^n n^{\gamma_d - 1}, & d = 1, 2, 3, d \geq 5 \\ \mathrm{A}_d \mu_d^n log(n)^{1/4}, & d = 4 \end{cases} \tag{22}$$

Inserting (22) when $d \geq 5$ into (3) yields

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

It is also known that $\gamma_d = 1$ for $d \geq 5$, which gives the following

$$A_d \mu_d^{n+m} \leq A_d^2 \mu_d^{n+m} \iff A_d \leq A_d^2 \iff 1 \leq |A_d|$$

Making the assumption that A_d is positive in the last step proves that the general bound (21) holds. It is intuitive that A_d has to be positive; if we study (22), we can see that $c_n(d)$ is dependent on A_d, μ_d^n and $n^{\gamma_d - 1}$ for $d \geq 5$. We know that the connective constant μ_d^n is positive from problem 7 and that $n \geq 1$. We also know that c_n, the number of possible self avoiding walks, cannot be negative. If A_d would be negative, we would get a negative c_n: an impossible result. Thus, we know that A_d is positive and it is clear that (21) holds.

1.9 Problem 9: Obtaining New Estimates

In the last problem of part 1, the objective is to estimate A_d, μ_d and γ_d for some $d \geq 3$ by using the SISR algorithm and the same technique as in problem

6. These estimates will then be compared to our results from problem 7 and 8 as well as with the following asymptotic bound for μ_d:

$$\mu_d \sim 2d - 1 - \frac{1}{2d} - \frac{3}{(2d)^2} - \frac{16}{(2d)^3} + O(\frac{1}{d^4}) \tag{23}$$

The chosen d for this problem is $d = 5$, which yields the following coefficient estimates and corresponding estimates for A_5, μ_5 and γ_5:

$$\beta_1 = .1228, \beta_2 = 2.178, \beta_3 = .0341$$

$$\log(A_5) = .1228 \iff A_5 = e^{.1228} \approx 1.13$$

$$n\log(\mu_5) = 2.178n \iff \mu_5 = e^{2.178} \approx 8.83$$

$$(\gamma_5 - 1)\log(n) = .0341\log(n) \iff \gamma_5 = 1 + .0341 \approx 1.03$$

Now that we have our estimates, we compare our estimate μ_5 with the general bound (19) from problem 7

$$d = 5 \leq \mu_5 = 8.83 \leq 2d - 1 = 9$$

As seen above, the estimation of μ_5 is inside our bound and our hypothesis that (19) holds for our SISR algorithm is confirmed for $d = 5$. Verifying that the estimate of A_5 is inside our general bound (21) is shown below

$$A_5 = 1.13 \geq 1$$

Thus, both μ_5 and A_5 satisfies their respective general bound. Finally, we estimate μ_5 with (23) and compare it with the estimation of μ_5 from the linear regression sampled from the SISR. The estimation of μ_5 with the asymptotic bound is presented below.

$$\mu_5 \sim 2 \cdot 5 - 1 - \frac{1}{2 \cdot 5} - \frac{3}{(2 \cdot 5)^2} - \frac{16}{(2 \cdot 5)^3} + O(\frac{1}{5^4}) \approx 8.85$$

As we can see, the result is similar to our estimation, with an error of about .02. This could be partly due to the error term $O(\frac{1}{5^4})$, which leads to a loss in information of the approximation of the asymptotic bound. It could also be that the estimation of μ_5 using our SISR algorithm is not complete, since we have a finite amount of samples ($N = 10^3$) and steps ($n = \{1, 2, ..., 100\}$). However, the estimates are close to each other, and due to the fact that we are operating with restrictions in computational power, we can conclude that the estimation precision from our SISR algorithm is good enough.

2 Filter Estimation of Noisy Population Measurements

2.1 Problem 10: Hidden Markov Model

In the last section of this home assignment we aim to estimate the relative population size for some organism in different generations k. The relative population

size is defined between zero and one, where zero means that the population is extinct, and one that it has reached its maximum capacity. The closer the population is to its maximum capacity, the more resources will be consumed, thus increasing the probability that the next generation will be smaller, and vice versa. In the following two parts, we model the relative population with the following transition distribution:

$$X_{k+1} = B_{k+1}X_k(1 - X_k), B_{k+1} \in U(\frac{1}{2}, 3), \text{iid}, k = \{0, 1, ..., 50\} \tag{24}$$

We estimate our initial distribution X_0 according to

$$X_0 \in U(\frac{1}{5}, \frac{3}{5}) \tag{25}$$

The measurement of the exact relative population is hard to determine. Hence, we use the following observation density to measure the relative population:

$$Y_k|X_k = x \in U(\frac{x}{2}, x) \tag{26}$$

The observation density Y_k can be seen as a hidden Markov model and the relative population size X_k as its corresponding hidden Markov chain. For our upcoming SISR implementation, the underlying measurements as well as the true values of the population size are stored in a file called "population.mat".

2.1.1 Part A: Estimating the Filter Expectation

Our objective in part A is to implement a SISR algorithm, like we did in problem 5, and then estimate the filter expectation τ_k for every generation k through the equation

$$\tau_k \approx \sum_{i=1}^{N} \frac{\omega_k^i}{\sum_{l=1}^{N} \omega_k^l} \phi(X_i^{0:k})$$

Applying the same technique described by Wiktorsson (2022, 24), using (26) as the observation density, (25) as our initial distribution and (24) as the transition distribution, we obtain our filter expectations for all generations k. The previously sampled Y_k's are collected from "population.mat", and the results from the SISR algorithm are displayed below in figure 5.

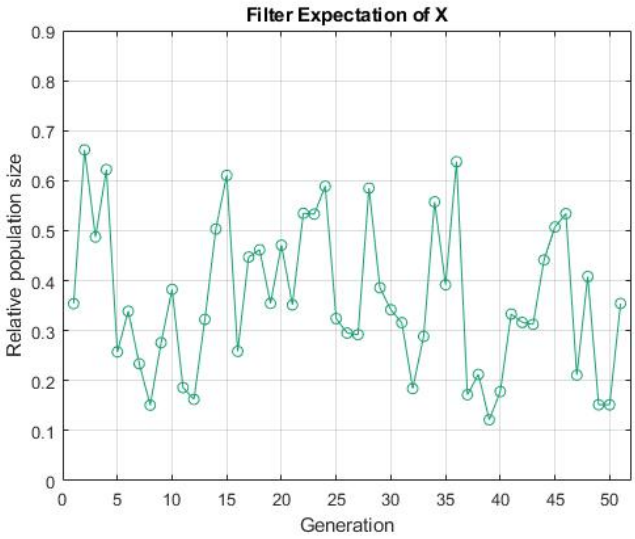

Figure 5: 2D plot of τ_k estimated from the SISR algorithm as k goes from 0 to 50. The green graph shows the point estimates of the relative population size for each generation k.

There are not too many interesting conclusions that can be drawn from figure 5, since its main purpose is to prepare for the latter part of problem 10. That being said, some insights can still be derived. The expectations of the relative population sizes satisfy $.1 < \tau_k < .7$, suggesting that the stipulation that all relative population sizes must be within the interval $I = [0, 1]$ holds for the algorithm. Furthermore, the mean value of all τ_k's is approximately equal to .36, and there is a pattern of regression to the mean when a τ_k is either unusually small or large. This indicates that the model captures the fact that when the population increases, more resources are consumed, thus affecting the relative population size of the next generation negatively, and vice versa. Moving on, we will now use the lower and upper bounds of our filter expectations presented in figure 5 to construct a 95% point wise confidence interval.

2.1.2 Part B: Point Wise Confidence Interval

In part B, we aim to create a 95% point wise confidence interval for all X_k's. Thereafter, in order to evaluate the accuracy of our filter expectations τ_k, we are to check how many X_k's that fall outside of the interval. Using the same technique as Wiktorsson (2022, 25) for constructing the confidence interval, combined with including the true values of X_k from "population.mat", yields the two graphs necessary for the analysis. These are presented in figure 6.

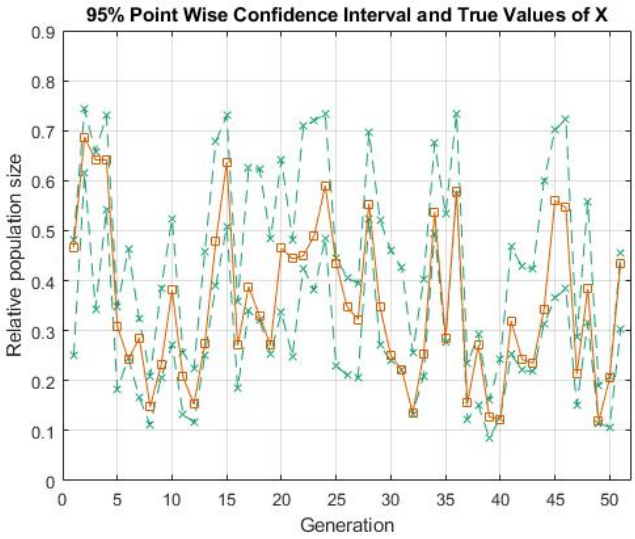

Figure 6: 2D plot of the approximated 95% confidence interval for the filter expectation τ_k at generation k compared to the true values of X_k. The green graph shows the lower and upper bound of the confidence interval, whilst the orange graph showcases the true relative population sizes.

When observing figure 6, the point wise confidence interval seems to be well constructed. We are using a 95% confidence level, and X_k falls outside the lower and upper limit for $k = \{5, 30, 39\}$. (It is to be noted that indexes in MATLAB start at 1 and not 0. Thus, the outliers in figure 6 are displayed at generation 6, 31, and 40.) A point wise confidence interval on a 95% confidence interval should, on average, fail to cover 5% of the true values. In our case, this corresponds to

$$.05 \cdot 51 \approx 2.55$$

observations. Hence, it is not problematic that X_5, X_{30}, and X_{39} are not covered by the interval. At last, we conclude that the τ_k's we obtained by adapting Wiktorsson's SISR algorithm are good estimates of the relative population size in generation k for the organism in question.

3 References

Slade, G 1994, 'Self-Avoiding Walks', *The Mathematical Intelligencer*, vol. 16, no. 1, pp. 31.

Wiktorsson, M 2022, *Sequential Monte Carlo methods III*, lecture slides, Monte Carlo methods for stochastic inference FMSN50, Lund University, pp. 16, 24-25, delivered 9 February 2022.