Lund University
Faculty of Engineering (LTH)

FMSN50

Monte Carlo and Empirical Methods for Stochastic Inference

Home Assignment 2

Authors:
Justin Ma (941123-7812), tfy14jma@student.lu.se
Erik Norlander (940601-7831), nat13eno@student.lu.se

The report was submitted on: February 20, 2018

Lunds universitet
Lunds Tekniska Högskola

Contents

Introduction

The essence of a self-avoiding walk (SAW) is simply a random walk that does not bite it's own tail. In this paper, we will study a SAW in a square lattice. There are numerous interesting applications of such walks within biology and engineering, many of which remain unsolved.

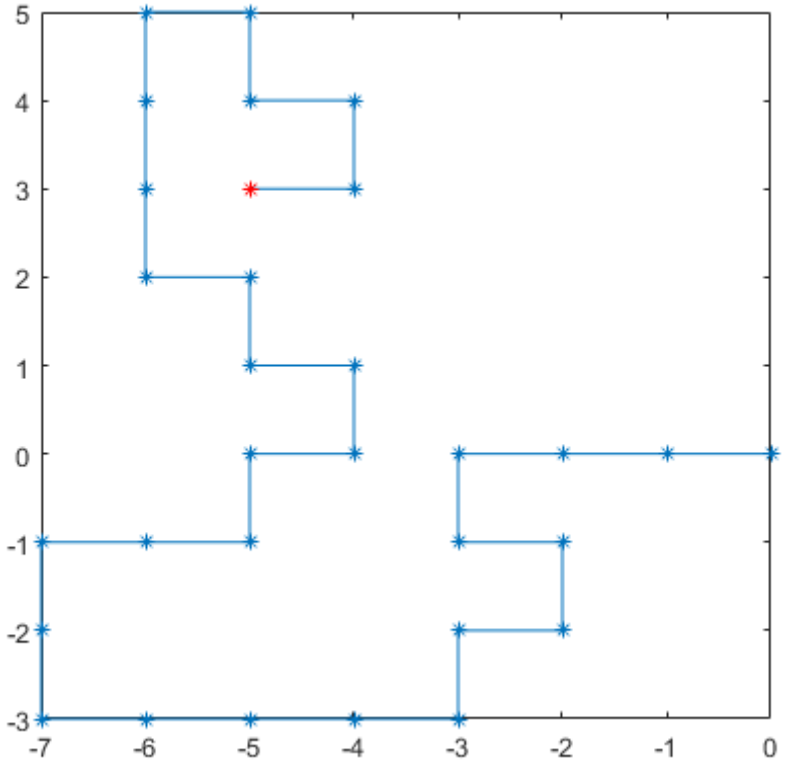

Figure 1: A self avoiding walk that failed after 30 steps.

A simple example of such a walk can be seen in figure 1 where the walk failed after 30 steps. A failure meaning that it found itself on a path where no new unvisited step was available.

We will investigate a few different methods of estimating the number of walks $c_n(d)$ for a given dimension d and a given step length n as well as some theoretical properties of it. The connective constant μ_d is also of great interest here as it can be interpreted as the geometric mean of the number of un-visited neighbours for a self-avoiding walk [1].

1 An important inequality

We denote the number of possible walks as $c_n(d) = |\boldsymbol{S}_n(d)|$, where $\boldsymbol{S}_n(d)$ is as defined in [1]. Then the inequality

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

follows because the right hand side describes all m number of ways that can be added on to an n-step SAW. This means that the right hand side includes the case $n+m$ as well as cases where they intersect [4].

2 Subadditivity and Fekete's lemma

Observing the inequality (1) we notice that it's logarithm is in fact *subadditive* as

$$\log(c_{n+m}(d)) \leq \log(c_n(d)) + \log(c_m(d)), \quad \Big[\log(c_n(d))\Big]_{n\geq 1}. \tag{2}$$

Therefore, by Fekete's lemma [1], the limit $\lim_{n\to\infty} \log(c_n(d))/n$ exist and is equal to $\inf_{n\geq 1} \log(c_n(d))/n$ which we denote as μ_d, the *connective constant*. From this we get [5]

$$\lim_{n\to\infty} \frac{1}{n} \log(c_n(d)) = \log(\mu_d) \leq \frac{1}{n} \log(c_n(d)). \tag{3}$$

Rearranging equation (3) and removing the logarithm yields the desired result as

$$\begin{aligned} e^{\log(\mu_d)} &= \lim_{n\to\infty} e^{\frac{1}{n}\log(c_n(d))} \iff \\ \mu_d &= \lim_{n\to\infty} c_n(d)^{\frac{1}{n}}. \end{aligned} \tag{4}$$

Estimating $\mathbf{c_n(d)}$ with Sequential Monte Carlo

In this section we want to estimate $c_n(d)$ using Sequential Monte Carlo methods (SMC's). Moreover, we are interested in estimating μ_d using relation (6), see [1]

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

We know that $c_n(d)$ can be estimated by considering the sequence $(f_n)_{n\geq 1}$ of uniform distributions on the set $(\boldsymbol{S}_n(d))_{n\geq 1}$ as equation (6), see [1]

$$f_n(x_{0:n}) = \frac{\mathbb{1}_{\boldsymbol{S}_n(d)}(x_{0:n})}{c_n(d)} \tag{6}$$

Comparing equation (6) with slide 5 in lecture 7, see [6], we realize that the indicator-function in this case is $z_n(x_{0:n})$ which is used in creating the weights ω_n

$$\omega_{n+1}^i = \frac{\mathbb{1}_{n+1}(X_i^{0:n+1})}{\mathbb{1}_n(X_i^{0:n}) g_{n+1}(X_i^{n+1} \mid X_i^{0:n})} \omega_n^i \tag{7}$$

From the same lecture we also learned that the procedure to estimate c_n with importance sampling boils down to the following equation (8):

$$c_n^{SIS} = \frac{1}{N} \sum_{i=1}^{N} \omega_n^i \tag{8}$$

One can improve on the SIS-model by using resampling. This means that you duplicate particles with large weights and kill particles with small weights [6], by randomly sampling from them according to their weights. A high number of neighbours yielding a large weight and vice versa. This is called Sequential Importance Sampling with resampling or SISR (9). The method has the same mean as SIS by the theorem found in the lecture notes [6, p.21] but adds some variance due to additional randomness.

$$c_n^{SISR} = \sum_{i=1}^{N} \frac{\omega_n^i}{\sum_{l=1}^{N} \omega_n^l} \phi(X_i^{0:n}) \tag{9}$$

Reference values for $c_n(2)$ for a square lattice when n = 1,2,...,39 was found [7] and is presented in figure 2.

Table 2. Values of c_n on the 2-dimensional square lattice. The most recent additions to this table are from [5].

n	c_n	n	c_n
1	4	21	2 408 806 028
2	12	22	6 444 560 484
3	36	23	17 266 613 812
4	100	24	46 146 397 316
5	284	25	123 481 354 908
6	780	26	329 712 786 220
7	2 172	27	881 317 491 628
8	5 916	28	2 351 378 582 244
9	16 268	29	6 279 396 229 332
10	44 100	30	16 741 957 935 348
11	120 292	31	44 673 816 630 956
12	324 932	32	119 034 997 913 020
13	881 500	33	317 406 598 267 076
14	2 374 444	34	845 279 074 648 708
15	6 416 596	35	2 252 534 077 759 844
16	17 245 332	36	5 995 740 499 124 412
17	46 466 676	37	15 968 852 281 708 724
18	124 658 732	38	42 486 750 758 210 044
19	335 116 620	39	113 101 676 587 853 932
20	897 697 164		

Figure 2: A reference table for values of c_n [7].

3 A naive approach: sampling from a random walk

First we want to simulate a random walk and count the amount of self avoiding $c_n(2)$ walks that's produced for the different step lengths n = 1,2,3.... Note we are only concerned about a 2D-model ($d = 2$) for problem 3-6. In order to do this we sample g_n from a completely random walk. This means to draw the next X_{k+1} uniformly from the four neighbors of X_k.

Because a random walk of this type always has 4 possible neighbors on a square lattice mean that the instrumental distribution function is uniformly $g_n = \frac{1}{4}$ for all neighbors. Therefore, using (7), the weights become

$$\omega_n = \begin{cases} 4^n & \text{if the walk is self avoiding} \\ 0 & \text{otherwise} \end{cases} . \tag{10}$$

Using equation (8) we realize a simple way to implement this form of sampling as

$$\begin{aligned} c_n = \frac{1}{N}\sum_{i=1}^{N}\omega_n^i = \frac{1}{N}\Big(N_{\text{Self Avoiding}} \cdot 4^n + N_{\text{Non-Self Avoiding}} \cdot 0\Big) = \\ = \frac{N_{SA}}{N} \cdot 4^n \end{aligned} . \tag{11}$$

Therefore, we simply have to compute the ratio $\frac{N_{SA}}{N}$ for a given step length n and multiply it with the weight to get c_n. With this approach the resulting c_n:s are: Theoretically, the first 3 values would be $c_{1,2,3}(2)$ = [4,12,36] if the walk is self avoiding. This is because the first point always has four possible neighbors, meaning $c_1(2) = 4$. The second point always has three possible neighbors and there's four different possible second points, meaning $c_2(2) = 4 \cdot 3 = c_1(2) \cdot 3 = 12$. Finally, the third point also always has three possible neighbors. Following the same logic yields $c_3(2) = 4 \cdot 3 \cdot 3 = c_2(2) \cdot 3 = 36$. After this it's more difficult to predict the amount of SAW:s and one should expect non-integer estimates that arises from averaging. However, from table 1 we realize that the only correct estimate of the first three is when $n = 1$, confirming that we can improve on this first naive attempt.

4 Sampling from a SAW with SIS

Now we try and estimate $c_n(2)$ when g_n is the distribution of actual self-avoiding walks. This means that g_n can not be considered to be $\frac{1}{4}$ anymore. It has to be calculated for each case $n > 3$. To get this we have to save the number of neighbors, $1/g_n$, in each iteration. Using the same SIS algorithm for the weights (7) we now get

$$\omega_{n+1} = \frac{\omega_n}{g_n} \quad . \tag{12}$$

Using this approach we get a result presented in table 1.

5 Sampling from a SAW with SISR

Here we want too implement the SISR-model. The scheme for SISR is as follows: we sample N-number of particles for a single step, then using the randsample-function in Matlab we can draw a new sample which is distributed according to the weights of each particle, which results in that particles with large weights gets duplicated and particles with small weights gets killed, we then repeat this for n-number of steps and yield the following result:

Table 1: In the following table we can see the values for our estimation of possible self-avoiding walks in a 2-dimensional plane for different n and for the three different methods that were presented above. These estimation were made with 1000 particles, running the code 10 times.

n	1	2	3	4	5	6	7	8	9	10
$c_n^{RW}(2)$	4	11.92	33.95	94.11	232.04	589.00	1559.76	4364.70	11219.76	27892.12
$c_n^{SIS}(2)$	4	12	36	100.19	285.01	784.15	2197.08	5945.83	16210.15	44243.06
$c_n^{SISR}(2)$	4	12	36	100.02	283.47	776.88	2150.06	5853.21	16091.11	43606.88

Comparing table 1 with figure 2 as a reference point for the c_n we see that the random walk is quite a hopeless estimate as n grows larger. SIS and SISR are both quite close to the reference values but it's a bit difficult to say with certainty which one is superior.

In order to compare the variances of SIS and SISR we used the MATLAB `var`-function and arrive and table 2 which shows SISR has an overall higher variance than SIS as suggested earlier. This is because the extra sampling steps each adds variance since there's more randomness in each one. Note that the variance for both methods when $n = 1,2,3$ is 0 because of the theoretical values presented in section 3.

Table 2: Variances of using SIS vs SISR.

n	1	2	3	4	5	6	7	8	9	10
$\mathrm{Var}(c_n^{SIS}(2))$	0	0	0	0.269	4.626	52.638	411.203	4978.439	25420.10	160707.01
$\mathrm{Var}(c_n^{SISR}(2)$	0	0	0	0.586	5.659	75.222	477.966	3776.463	37287.75	463096.80

6 Estimates of A_2, μ_2 and γ_2

To estimate A_2, μ_2 and γ_2 we investigate (1) for $d = 2$. Taking the logarithm on both sides yields the following system of equations

$$\begin{cases} \ln(c_1) &= \ln(A_2) + \ln(\mu_2) + (\gamma_2 - 1)\ln(1) \\ \ln(c_2) &= \ln(A_2) + 2 \cdot \ln(\mu_2) + (\gamma_2 - 1)\ln(2) \\ \vdots & \quad \vdots \\ \ln(c_k) &= \ln(A_2) + n \cdot \ln(\mu_2) + (\gamma_2 - 1)\ln(k) \end{cases} \tag{13}$$

Which can easily be rewritten on a matrix notation for simplicity of computation. Considering the general case of $\ln(c_n)$ and rearranging leaves us with:

$$\ln(c_n) + \ln(n) = \ln(A_2) + n \cdot \ln(\mu_2) + \gamma_2 \ln(n) \tag{14}$$

We redefine this problem as a multiple linear regression model so $\alpha_n = \ln(c_n) + \ln(n), n \in [1,...k]$, $\beta_1 = \ln(A_2)$, $\beta_2 = \ln(\mu_2)$ and $\beta_3 = \gamma_2$ so

$$\alpha = [\alpha_1, \alpha_2, ..., \alpha_k]^T \quad \beta = [\beta_1, \beta_2, \beta_3]^T \quad X = \begin{bmatrix} 1 & 1 & \ln(1) \\ 1 & 2 & \ln(2) \\ \vdots & \vdots & \vdots \\ 1 & k & \ln(k) \end{bmatrix} \tag{15}$$

Where $\alpha = X\beta$. Solving for β yields the estimate

$$\hat{\beta} = (X^T X)^{-1} X^T \alpha \tag{16}$$

where

$$\hat{A}_2 = \exp(\hat{\beta}_1) \quad \hat{\mu}_2 = \exp(\hat{\beta}_2) \quad \hat{\gamma}_2 = \hat{\beta}_3 \tag{17}$$

We also have to study the variance of this estimate to determine which one is the most difficult to estimate. However, the normal variance equations for MLR does not apply here as they generally assume independence. Since c_n is not independent for n = 1,2,... we have to use the built in MATLAB functions instead.

Table 3: Estimates of A_2, μ_2 and γ_2 where $n = 10$. The simulation was run 10 times with 1000 particles.

Estimate	Mean	Variance (10^{-5})
A_2	1.4873	4.30
μ_2	2.6691	24.21
γ_2	1.2074	20.09

Since A_2 has the smallest variance it is the easiest to estimate of the constants. From the assignment [1] we know that $\gamma_2 = 43/32 = 1.3438$ which is quite close to γ_2 in table 3. However, not within our margin of error.

We know from (4) that μ_d approaches $c_d(n)^{\frac{1}{n}}$ for large n. This can be verified by testing for a large n, we chose $n = 50$. $c_{50}(2) = 4.5586 \cdot 10^{21}$ according to our calculations (however, there is of course variability in this result) and $c_{50}(2)^{\frac{1}{50}} = 2.7113$ which is close to our estimate of μ_2. In figure 3 we can see how the estimate (4) converges for large n.

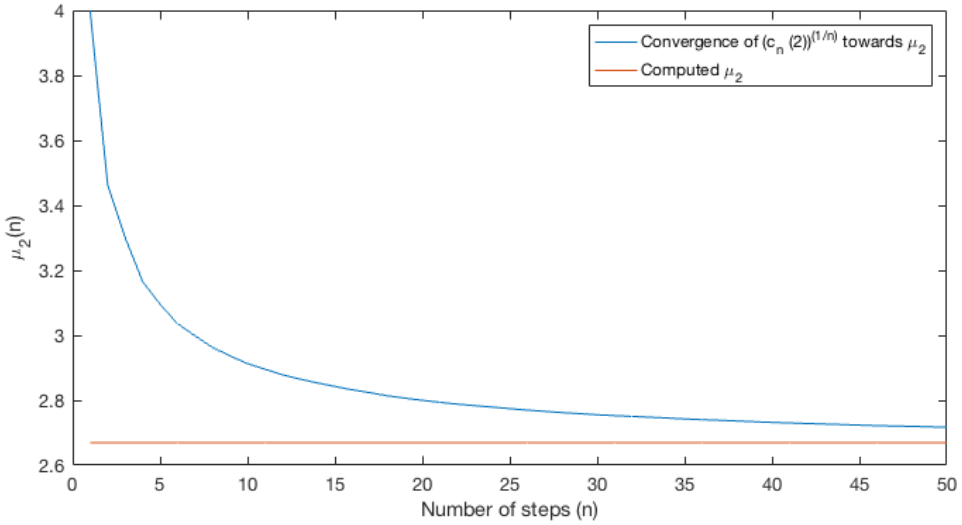

Figure 3: Displaying how $c_n(d)^{\frac{1}{n}}$ approaches our computed μ_2 for high n.

7 A general bound of μ_d

Let's investigate the upper and lower bounds of c_n. The smallest possible amount of steps for a self avoiding walk would be a walk which only takes steps in a positive direction. Meaning we have a lower bound of d^n steps. The set of all walks having no reversals includes all self-avoiding walks. The first step of such a walk has $2d$ choices while the subsequent steps has $2d-1$. Meaning, we have an upper bound of $2d(2d-1)^{n-1}$, see [7]

$$d^n \leq c_n \leq 2d(2d-1)^{n-1}. \tag{18}$$

Remembering equation (4) it's clear that $\mu_d^n \leq c_n$. Therefore,

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

$$\left(\frac{2d}{2d-1}\right)^{\frac{1}{n}}(2d-1) = 2d-1, \quad \text{as } n \to \infty \quad \text{meaning}$$

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

8 A general bound of A_d

We want to show that

$$A_d \geq 1, \quad d \geq 5. \tag{20}$$

Following the hint we use equations (1) and (5) which gives us

$$\begin{cases} c_{n+m}(d) & \leq c_n(d)c_m(d) \\ c_n(d) & = A_d\mu_d^n n^{\gamma_d - 1}, \quad d \geq 5, \gamma_d = 1 \end{cases} \tag{21}$$

$A_d\mu_d^{n+m}(n+m)^{\gamma_d-1} \leq A_d\mu_d^n n^{\gamma_d-1} A_d\mu_d^m m^{\gamma_d-1} = A_d^2\mu_d^{n+m}n^{\gamma_d-1}m^{\gamma_d-1}, \quad \gamma_d = 1 \Longrightarrow$
$A_d\mu_d^{n+m}(n+m)^0 \leq A_d^2\mu_d^{n+m}n^0m^0 \Longrightarrow A_d\mu_d^{n+m} \leq A_d^2\mu_d^{n+m} \Longrightarrow A_d \leq A_d^2$
meaning $\quad A_d \geq 1$

9 Estimating A_d, μ_d and γ_d for $d \geq 3$

Using the same technique as in section 6 by identifying a linear regression model we estimate the constants for higher dimensions.

Table 4: In the following table we can see the estimate for A_d, μ_d and γ_d, using SISR, for $d = 3, 5, 10$. The simulation was run 10 times with 1000 particles

Estimate	Mean	Variance (10^{-5})
A_3	1.2655	1.46
A_5	1.1308	0.13
A_{10}	1.0569	0.01
μ_3	4.7177	28.48
μ_5	8.8275	13.36
μ_{10}	18.925	4.05
γ_3	1.1075	8.09
γ_5	1.0351	1.50
γ_{10}	1.0070	0.08

Considering the bound (19) on μ_d it is clear for all our values μ_d as

$$3 < \mu_3 < 5, \quad 5 < \mu_5 < 9, \quad 10 < \mu_{10} < 19.$$

Moreover, observing 4 it is clear that the bound (20) applies to all our values of A_d since they are all larger than 1. However, it is creeping closer to 1 as d grows. It seems as if the estimates of γ_d becomes more accurate as d grows because it also creeps closer to 1 which, of course, is the theoretical value.

Regarding μ_d we have a asymptotic bound from Graham given in [1]

$$\mu_d \sim 2d - 1 - 1/(2d) - 3/(2d)^2 - 16/(2d)^3 + O(1/d^4) \tag{22}$$

predicts our estimates of μ_d very closely for these selected values:

$$\begin{aligned} \text{Graham:} \quad & \mu_3 = 4.6759 \quad \mu_5 = 8.8540 \quad \mu_{10} = 18.941 \\ \text{Estimate:} \quad & \mu_3 = 4.7172 \quad \mu_5 = 8.8275 \quad \mu_{10} = 18.925 \end{aligned}$$

Furthermore, we display how close our estimates are to the asymptotic bound by presenting figure 4. Of course we can not say that all our estimates would fit the asymptotic bound this well, but what is clear is that our estimates for d = 3,5,10 of μ_d are very suitable.

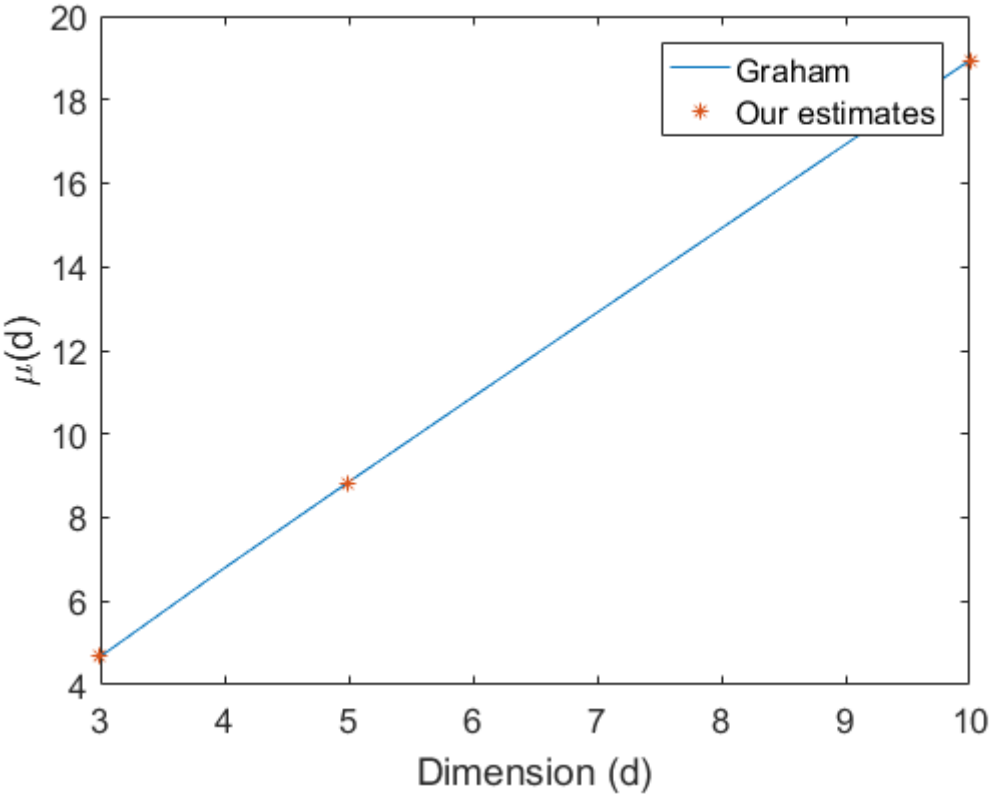

Figure 4: The asymptotic bound (22) and our estimates.

Finally, in table 4 we observe that the variance of our estimates decrease as d grows. Considering the asymptotic bound from Graham (22) this is natural for μ_d as d becomes larger because the error term $O(1/d^4)$ becomes smaller quickly as d grows which is an interesting property of the connective constant.

Conclusion

The aim of this home assignment was to solve the complex combinatoric problem of computing the number of possible self-avoiding walks $c_n(d)$. Using sequential Monte Carlo methods we have estimated the number $c_n(d)$ for different steps n and dimensions d. As we can see in table 1 the number of such possible walks grows very quickly as the step size n grows which would be expected.

Another point of interest is the behaviour of the connective constant μ_d; more specifically, its asymptotic behaviour. We were able to both verify that the limit (4) holds as n grows larger for a set d and the asymptotic bound (22) holds as d grows larger and n remains constant. Interestingly, its variance also became smaller for larger d, following the bound (22) closely. Not only did the asymptotics hold, but the bound (19) also did.

Estimates of the other two constants A_d and γ_d also produced promising results.

A_d kept above 1 for all d's that were tried and γ_d approached 1 as d became higher.

It's also possible to see the difference in variance of the methods, where the SISR-method has a higher variance compared to SIS as it introduces more randomness.

References

[1] *Home assignment 2*

[2] *The connective constant of the honeycomb lattice equals* $\sqrt{2+\sqrt{2}}$

[3] *Borel type bounds for the self-avoiding walk*

[4] *The Self-Avoiding Walk: A Brief Survey*

[5] *Lectures on Self-Avoiding Walks*

[6] *Lecture 7 in FMSN50*

[7] *Self-avoiding walks*