The Rosenbluth and the pruned-enriched Rosenbluth algorithm for simulating polymers on a lattice

Sajag Kumar *

School of Physical Sciences (SPS),
National Institute of Science Education and Research (NISER) Bhubaneswar

We begin with the simulation of a random walk on $\mathbb{Z}_2$ and $\mathbb{Z}_3$. Then we describe a model of polymers as self-avoiding walks on a lattice. We then simulate SAWs on $\mathbb{Z}_2$ and $\mathbb{Z}_3$ using the Rosenbluth method and compute the average end-to-end distance of the polymer. We further implement the pruned-enriched Rosenbluth method, which is a modified version of the Rosenbluth method.

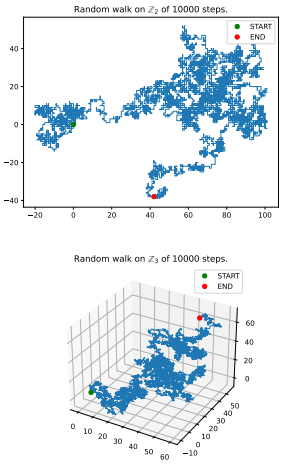

Figure 1: Random walks of 10000 steps on $\mathbb{Z}_2$ and $\mathbb{Z}_3$

I. RANDOM WALKS ON LATTICE

We have discussed random walks in class. Generating them is among the simplest applications of the Monte Carlo methods. The random walk we have implemented here is slightly different from the one we discussed in class. The difference being our random walk is constrained to move on a lattice. The step size is always one. In particular we have simulated random walk on $\mathbb{Z}_2$ and $\mathbb{Z}_3$.

II. POLYMERS AS SELF-AVOIDING RANDOM WALKS

During a random walk a point may be revisited. A self-avoiding random walk (SAW) does not visit the point it has already visited. SAWs form a simplistic yet useful toy models for polymers. In particular, SAWs form a very good model for polymers that are linear and unbranched. Following are some important points about the model:

1. The polymer is assumed to be made up of monomers of equal length. These monomers are the steps in the SAW.

2. The end points of the polymers interact via the Lennard-Jones potential. But we switch this potential off.

3. The Lennard-Jones potential is switched off but the two end points of the polymer is assumed to repel each other. This is a valid assumption when the polymers are present in some solvent.

4. We study the polymers on lattice, which further restrict the position of the monomers with respect to each other but large scale behaviour is not sensitive to this.

The squared end-to-end distance of a polymer of length N on $\mathbb{Z}_2$ scales as

$$< r^2 > \sim \ N^{\frac{6}{4}}$$

and on $\mathbb{Z}_3$

$$< r^2 > \sim \ N^{\frac{6}{5}}.$$

A. Generating SAWs

We have implemented the one-step-look-ahead method for generating SAWs on both $\mathbb{Z}_2$ and $\mathbb{Z}_3$. Following are the steps involved in the algorithm:

1. The walk begins at the origin.

2. After each step the algorithm looks for neighbouring sites that are unoccupied on the lattice.

* sajag.kumar@niser.ac.in

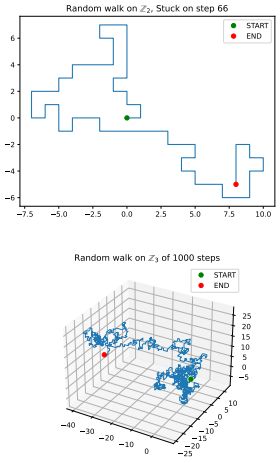

Figure 2: An example of generating SAW. The SAW on $\mathbb{Z}_2$ got stuck on the 66th step, while on $\mathbb{Z}_3$ did not stuck and completed 1000 steps.

3. The algorithm chooses one among the available sites at random. Each available site is equally likely to be chosen.

4. The algorithm stops when either the number of steps have been reached or when there are no unoccupied neighbouring sites.

The problem with this algorithm is that we are not guaranteed a SAW of desired number of step. The walk may get stuck after some steps. Also some of the walks are more probable than others. All of this implies that the estimate for the squared end-to-end distance for the polymers would not be good.

III. ROSENBLUTH METHOD

This is an importance sampling approach to estimate the squared end-to-end distance of a polymer on a lattice. The SAWs generated through the one-step-look-ahead method form the trial function. The method is as follows:

1. The SAW begins at the origin. At each step (n) of the walk we assign a weight W_n to the polymer. $(W_1 = 1)$

2. At any step m, the weight is modified as follows:

   (a) If there are no unoccupied neighbours

   $$W_m = 0.$$

Table I: Comparison of end-to-end distance as computed through Rosenbluth method and the expected $N^{6/4}$ and $N^{6/5}$ for $\mathbb{Z}_2$ and $\mathbb{Z}_3$ respectively. It is clearly visible that for larger values of N (number of steps) the Rosenbluth gave bad values. The values in case of $\mathbb{Z}_2$ are much worse than in case of $\mathbb{Z}_3$ because the number of walks that stuck in $\mathbb{Z}_3$ are less than in $\mathbb{Z}_2$ at smaller values. This happens because $\mathbb{Z}_3$ has an additional degree of freedom.

N	$< r^2 >_{2D}$	$N^{\frac{6}{4}}$	$< r^2 >_{3D}$	$N^{\frac{6}{5}}$
10	29.9	31.6	18.7	15.8
20	77.3	89.4	40.6	36.4
30	133.7	164.3	64.2	59.2
40	202.1	253	91.1	83.7
50	281.2	353.6	121.6	109.3
60	387.6	464.8	145.3	136.1
70	422.8	585.7	176.5	163.7
80	546.5	715.5	210.4	192.2
90	612.3	853.8	236.6	221.4
100	769.1	1000	266.9	251.2
110	821.6	1153.7	318.2	281.6
120	1172.8	1314.5	370.4	312.6
130	492.8	1482.2	377.9	344.1
140	443.5	1656.5	380.8	376.1
150	277.8	1837.1	406.7	408.6

   (b) If there are a unoccupied neighbours and b neighbours in total

   $$W_m = \frac{a}{b-1} W_{m-1},$$

   here the denominator is $b-1$ because one of the neighbours is always occupied as the polymer must have reached the present site through a neighbour. b is 4 for $\mathbb{Z}_2$ and 6 for $\mathbb{Z}_3$.

3. We generate N polymers of the same length (L) independently in the method as described above and the compute the weighted average of the squared end-to-end distance as follows:

   $$< r^2 >= \frac{\sum r_{(i)}^2 w_{(i)}}{\sum w_{(i)}},$$

   where $r_{(i)}^2$ is the squared end-to-end distance of the i-th polymer generated by the method.

The Rosenbluth algorithm runs into trouble when the length of the polymers is large. The number of polymers of desired length that it can produce is significantly low. A few polymers and their weights dominate the average.

IV. THE PRUNED-ENRICHED ROSENBLUTH METHOD

Table II: Comparison of end-to-end distance as computed through PERM and the expected $N^{6/4}$ and $N^{6/5}$ for $\mathbb{Z}_2$ and $\mathbb{Z}_3$ respectively. The value through the PERM algorithm at higher values of N for the 2D case is much better than Rosenbluth. For the 3D case Rosenbluth performed better but for larger values of N (> 5000) PERM will perform better. At smaller values of N (that is those values at which most of the SAWs do not get stuck) there is not much difference in the performance of PERM and Rosenbluth (PERM even performed worse than Rosembluth in the 3D case).

N	$< r^2 >_{2D}$	$N^{\frac{6}{4}}$	$< r^2 >_{3D}$	$N^{\frac{6}{5}}$
10	36.3	31.6	8.3	15.8
20	84.6	89.4	15.1	36.4
30	148.5	164.3	21.9	59.2
40	232.8	253	28.9	83.7
50	294.6	353.6	35.8	109.3
60	388.5	464.8	42.3	136.1
70	470.1	585.7	47.8	163.7
80	604.2	715.5	54.8	192.2
90	657.7	853.8	62.6	221.4
100	697	1000	69	251.2
110	793.6	1153.7	75	281.6
120	1021.9	1314.5	82.2	312.6
130	836.9	1482.2	87.3	344.1
140	1398.6	1656.5	94	376.1
150	3141.6	1837.1	101.7	408.6

The PERM algorithm helps in overcoming the imbalance of weights that is developed in the Rosenbluth algorithm. The algorithm has a pruning step where it prunes the effect of low weight polymers on the average and an enriching step where it enriches the effect of polymers of higher weights on the average. The pruning and enriching steps are as follows, suppose the length of the subunit of the polymer in the Rosenbluth algorithm is L with weight w then:

1. *Pruning.* If $w \leq W_-$ then one of the following steps is performed at random.
   (a) The polymer is discarded.
   (b) The weight is doubled.
2. *Enriching.* If $w \geq W_+$ then the weight of the polymer is halved and a copy of the polymer is added with equal weight.

The constants W_- and W_+ are initially set to 0 and 10^{100} respectively. And after each step their value is changed to w and $10w$ respectively. I had difficulty in understanding the origin of the constants, so I went with what the original paper suggested. They have also mentioned that the output of PERM is quite insensitive to these constants.

V. CONCLUSIONS AND DISCUSSIONS

We have implemented the Rosenbluth and the PERM algorithm for the estimation of end-to-end distance of polymers on square and cubic lattice. The superiority of PERM over Rosenbluth method is argued and demonstrated.

The initial plan was to plot the findings and verify the scaling law for the lattices through least square fitting. But for plots to be relevant we had to plot many points which required much more computational power than a laptop. Even the scripts which are trying to compute only fifteen points take more than 20 minutes to run. Another computational hindrance is not being able to call the Rosenbluth or the PERM function on step sizes of the order of thousands.

The superiority of the PERM algorithm even with number of steps currently available could be even more apparent if we could average over large number of polymers for each N. At present we are averaging over 10000 polymers for each step, taking this number higher would show the vulnerabilities of Rosenbluth and the superiority of PERM.

Our analysis for the cubic lattice has been for small N, SAWs on $\mathbb{Z}_3$ usually do not stuck for so few steps. This happens because of the extra degree of freedom that the cubic lattice has, on square lattice the maximum number of available sites is 3 while on a cubic lattice it is 6, so a SAW on a square lattice is more likely to get stuck quickly. Thats why the results from cubic lattice are anomalous. For high values of N, the superiority of PERM would be much more apparent.

[1] Computational Physics, *Jos M. Thijssen.*, 2nd Edition, Cambridge University Press.

[2] Monte Carlo Calculation of the Average Extension of Molecular Chains, *Marshall N. Rosenbluth and Arianna W. Rosenbluth*, J. Chem. Phys. 23, 356 (1955).

[3] Pruned-enriched Rosenbluth method: Simulations of θ polymers of chain length up to 1000000, *Peter Grassberger*, Phys. Rev. E, Vol. 56, No.3 (1997).

[4] Conputaitonal Physics, *TU Delft*, Quantum Tinkerer, `https://compphys.quantumtinkerer.tudelft.nl`.