Monte Carlo Simulation of Polymers

Simulating polymers as self-avoiding random walks

Francke, A. (4713672)
Glorie, M. (4279492)
Sangers, J. (4645197)
Applied Physics - Computational Physics
TU Delft

05-10-2022

Abstract

In this paper two characteristics of polymers grown on a 2 dimensional square grid were investigated with the help of Monte-Carlo methods. The polymers were generated according to the Rosenbluth method and Pruned-enriched Rosenbluth method (PERM) described in [1] and [2] respectively. After which through approximate importance sampling the average end-to-end distance and radius of convergence were determined as a function of the polymer length. Both methods showed results which scaled slower than expected from literature with the difference in critical exponent being between 8% and 16%. Furthermore the performance of both methods was compared which showed a decrease in computation time for generating a representative data-set when using PERM which comes at the cost of an increase in polymer correlation.

Contents

1 Introduction

When the dimensionality of a problem increases it often becomes computationally infeasible to use standard numerical integration methods to evaluate the physical characteristics of a system. Here, the use of Monte-Carlo methods can prove extremely useful in reducing the computational complexity while maintaining a high level of accuracy. In this report we apply approximate importance sampling in order to compute the end-to-end distance and radius of gyration of polymers modeled as self avoiding walks as a function of the polymer length.

In chapter 2 an overview of the theory behind the project is given, as well as an explanation of the sampling method used, which is described by Rosenbluth and Rosenbluth in [1]. It also introduces an enhanced version of this method, which was proposed by Grassberger in [2] and explains how the results we obtained were analysed to compare the two methods. Chapter 3 shows and discusses the results of our simulations, after which our conclusions and recommendations for further research are laid out in chapter 4.

This project was performed as part of the Computational Physics course at the Technical University of Delft.

2 Methods

2.1 Polymer characterisation

The polymers that are generated in this project are modeled as self-avoiding random walks (SAW) on a lattice. We use a two-dimensional square lattice where the distance between lattice points can be set to 1 without loss of generality. A polymer is described by an ordered set of points $\mathbf{r}_0, \mathbf{r}_1, ..., \mathbf{r}_L$, where each $\mathbf{r}_i$ describes the location of the i-th monomer in the polymer chain. The behaviour of two interesting characteristics of these polymers is investigated as a function of their length L; the end-to-end distance ($r_e^2(L)$) and the radius of gyration ($r_g^2(L)$).
End-to-end distance:

$$r_e^2(L) = |\mathbf{r}_L - \mathbf{r}_0|^2 \tag{1}$$

Radius of gyration:

$$r_g^2(L) = \frac{1}{L+1} \sum_{i=0}^{L} |\mathbf{r}_i - \mathbf{r}_{\text{cm}}|^2 \tag{2}$$

with $r_{\text{cm}} = \frac{1}{L+1} \sum_{i=0}^{L} \mathbf{r}_i$ the center of mass of the polymer.

The end-to-end distance and radius of gyration obey the scaling law [3], which is shown in equation (3) and (4) for the end-to-end distance and radius of gyration respectively.

$$\langle r_e^2(L) \rangle \sim L^{2v} \tag{3}$$

$$\langle r_g^2(L) \rangle \sim L^{2v} \tag{4}$$

For a SAW in two spacial dimensions the critical exponent v is equal to $\frac{3}{4}$, unlike the free random walk which has a critical exponent of $\frac{1}{2}$.

2.2 Sampling method

In order to compute these average quantities, we compute a set of random polymers according to the method described by Rosenbluth[1]. This method grows the polymers one monomer at a time avoiding any lattice site which has already been occupied by the polymer, as opposed to generating a set of free random walks and discarding all walks which are not SAW, which decreases quickly in terms of computational efficiency as the length of the polymers increases. The Rosenbluth algorithm is much less computationally expensive; however, not all SAW have the same probability of being generated and as such we need to sample the set of polymers accordingly in order to accurately estimate the end-to-end distance and radius of gyration.
To this end each polymer k is assigned a weight, $w_k^{(L)}$ during the generation process with which we compute the average end-to-end distance or radius of gyration as a weighted average[3]:

$$\langle r^2 \rangle = \frac{\sum_{k=1}^{N} w_k^{(L)} r_k^2(L)}{\sum_{k=1}^{N} w_k^{(L)}} \tag{5}$$

These weight are computed by recording the number of available positions, $m_{k,i}$ the polymer has to grow at each growing step from which the resulting weight is computed as:

$$w_k^{(L)} = \prod_{i=1}^{L} m_{k,i} \tag{6}$$

To investigate the dependence of the end-to-end distance and radius of gyration on the length of the polymers in one simulation we grow a set of random polymers up to a certain length L, these polymers can then be used to compute the average end-to-end and radius of gyration for polymers of all lengths up to and including L. However, a problem arises when looking at polymers of higher lengths; since the probability of a polymer getting stuck increases with the length of the polymers our data-set will always contain a larger amount of polymers of smaller lengths than polymers of length L. This in combination with the fact that the large differences in weights will mean the weighted average is dominated by a small set of polymers which causes the error in the polymer characteristics to increase significantly for large polymers.

2.3 PERM algorithm

To solve this, we implement an enhanced version of the Rosenbluth method called the Pruned-enriched Rosenbluth method (PERM) as proposed by Grassberger[2]. This method prunes and enriches the set of polymers during the growing process through implementation of parameters c_+ and c_-. The ratio of these parameters is set to $c_+/c_- = 10$ as suggested in [2], while the value chosen for one of the parameters determines how much the data-set is pruned and enriched. The algorithm works by calculating the average weight over all the polymers at length L', denoted as $W^{L'}$, and either pruning or doubling the weight of polymers which have a weight smaller than c_-W^L, each occurring with equal probability; or enriching by creating a copy of polymers with a weight larger than $c_+W^{L'}$ and consequently halving the weight of both the original and the copy. This results in no change in the average of the polymer characteristics, but it does have the effect of reducing the number of polymers that contribute little to the weighted average while increasing the number of polymers that contribute heavily. One has to be careful when choosing the value of c_+ since choosing a value that is too low will result in an over-enriched data-set in which we lose the benefit of statistically independent polymers, which will again lead to an increase in error.

2.4 Polymer correlation estimate

The previously discussed pruning and enriching steps from Grassberger's method whose frequencies are governed by the factor c_+ are said to influence the correlation between the polymers created using this PERM method.[2] [3] To test this claim we propose the following method for finding a representative value for the correlation between all polymers within an experimental run of the simulation.
All polymers created during an experimental run are fully characterised by the angles every monomer in the polymer makes with respect to the global orientation of the axis in the simulation. Since we simulated the polymers in two dimensions there are four possible orientations for every monomer, a polymer made up of $\mathcal{N}$ monomers is then fully characterised by a series of angles $s \in \Omega^{\mathcal{N}}$ where $\Omega = \{0, 1, 2, 3\}$ with $0, 1, 2, 3$ the integer multiples of 90° counter-clockwise from the positive x-axis. To determine the correlation between two polymers their first monomers are aligned, then the polymers i and j with angles characterising the polymers s_i and s_j two more discrete signals x_i and y_i are created where the monomers aligned, anti-aligned, and not aligned, in the respective direction, are encoded as +1,0,−1 respectively. The correlation between two polymers i and j is then calculated by equations (7) in which the bar denotes the mean of the realisation. An example of this method is included in the appendix. Performing this method for all N polymers in an experimental run yields us a $N \times N$ matrix with values $r^{i,j}$, this matrix' lower triangular part is then averaged

to a single metric for the correlation of all polymers, $\tilde{\mathcal{C}}$.

$$r^{i,j} = \sqrt{\left(r_x^{i,j}\right)^2 + \left(r_y^{i,j}\right)^2} \tag{7}$$

$$r_x^{i,j} = \frac{\sum_{k=1}^{\mathcal{N}} \left(x_{i,k} - \bar{x_i}\right)\left(x_{j,k} - \bar{x_j}\right)}{\sqrt{\sum_{k=1}^{\mathcal{N}} \left(x_{i,k} - \bar{x_i}\right)^2 \sum_{k=1}^{\mathcal{N}} \left(x_{j,k} - \bar{x_j}\right)^2}} \tag{8}$$

2.5 Error of the weighted averages

In order to determine the error of the weighted averages we apply the bootstrapping method. During our simulation a set of N polymers and their weights is generated. From this set N polymers are randomly drawn to form a new set of N polymers, with the possibility that a polymer can be drawn multiple times or not at all. This process is repeated n times to form n sets of N polymers. For each of these sets the weighted average (equation (5)) of the end-to-end distance and radius of gyration is computed and used to calculate the error of the original set using equation (9).

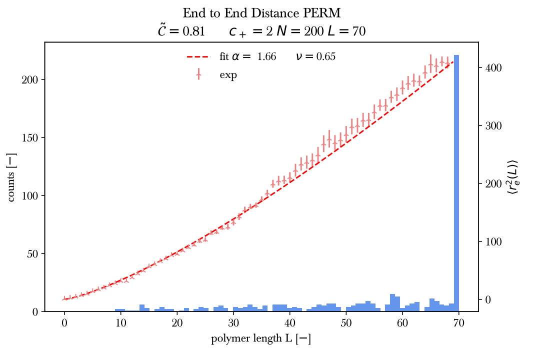
$$s(\langle r^2(L)\rangle) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\langle r^2(L)\rangle_i\right)^2 - \left(\frac{1}{n}\sum_{i=1}^{n}\langle r^2(L)\rangle_i\right)^2} \tag{9}$$

3 Results

3.1 End-to-end distances for Rosenbluth and PERM

The end-to-end distances for PERM and Rosenbluth are graphed in Figure 1 and Figure 2, respectively. For PERM, a c_+ value of 2 was chosen since this introduced the least extra correlation and provided a sufficient decrease in computation time; the function started with $N = 200$ polymers, which it tried to grow to a length of $L = 70$. Fitting to PERM's data values with a LSQ fitting function yielded an exponent $\nu = 0.65$ which is lower than the 0.75 expected for two-dimensional systems [3].

For the Rosenbluth method we had to generate significantly more polymers such that the shapes of long polymers were equally represented in the data, we therefore chose to generate $N = 800$ polymers that all had to reach length $L = 50$ all previous attempts were also included in the data. The need for more polymers the longer the chain length made us settle on $L = 50$ for this simulation since it would otherwise have taken to much time to generate the polymers. The result of fitting the data to the scaling law was a value of $\nu = 0.63$ which is less than PERM's result and deviates even further from the previous literature.



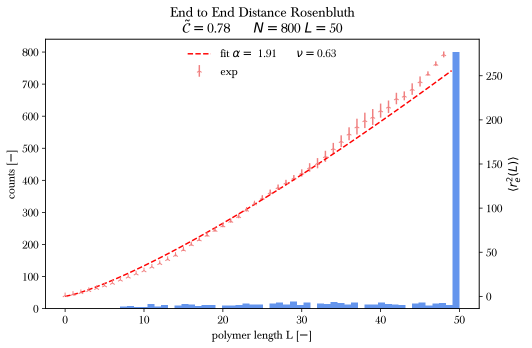
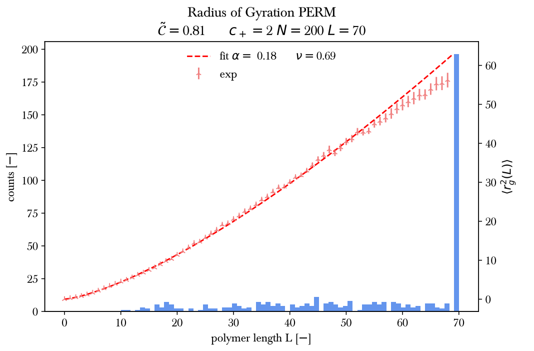


Figure 1: In blue a histogram of polymer counts with end lengths L' is shown. In light red the computed end-to-end distances are graphed with their error values and in dark red is a least squares function estimate of the parameters α and ν such that the LSQ error between the data set and the function $r^2_{e,est}(L) = \alpha L^{2\nu}$ is minimised. The data set generated 200 polymers up to length $L = 70$ using PERM with a c_+ value of two.

Figure 2: In blue a histogram of polymer counts with length L' is shown. In light red the computed end-to-end distances are graphed with their error values and in dark red is a least squares function estimate of the parameter α and ν such that the LSQ error between the data set and the function $r^2_{e,est}(L) = \alpha L^{2\nu}$ is minimised. The data set generated 800 polymers up to length $L = 50$ using the Rosenbluth method.

3.2 Radii of gyration for Rosenbluth and PERM

The dependence of the radius of gyration on the polymer length has been graphed for data gathered by PERM and the Rosenbluth method in Figure 3 and Figure 4 respectively. The data for these graphs was generated the same way as for the results in 3.1. Here we find values for the exponent of the scaling law equal to $\nu_{PERM} = 0.69$ and $\nu_R = 0.68$ for the PERM and Rosenbluth method respectively. Both these values are lower than the literature value of $\nu = 3/4$ for two dimensional systems. [2] [1] [3]



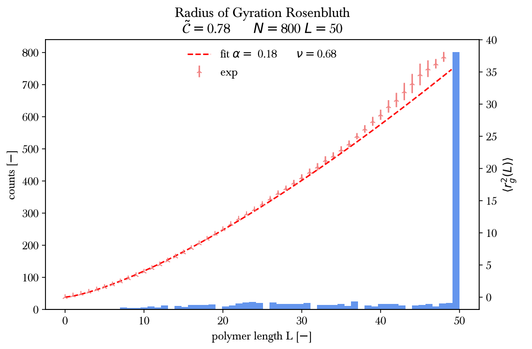


Figure 3: In blue a histogram of polymer counts with length L' is shown. In light red the computed radii of gyration are graphed with their error values and in dark red is a least squares function estimate of the parameter α and ν such that the LSQ error between the data set and the function $r^2_{g,est}(L) = \alpha L^{2\nu}$ is minimised. The data set generated 200 polymers up to length $L = 70$ using PERM with a c_+ value of two.

Figure 4: In blue a histogram of polymer counts with length L' is shown. In light red the computed radii of gyration are graphed with their error values and in dark red is a least squares function estimate of the parameter α and ν such that the LSQ error between the data set and the function $r^2_{g,est}(L) = \alpha L^{2\nu}$ is minimised. The data set generated 800 polymers up to length $L = 50$ using the Rosenbluth method.

3.3 Free random walk

For contrast the end-to-end distance and radius of gyration were also calculated for a data-set of polymer grown with crossing and backtracking allowed, modelled as free random walks (FRW). Since these polymers never get stuck we can easily compute a large data-set of long polymers; a data-set of 10000 FRW polymers was generated from which the average end-to-end distance and radius of gyration were calculated and are plotted as a function of the polymer length below.

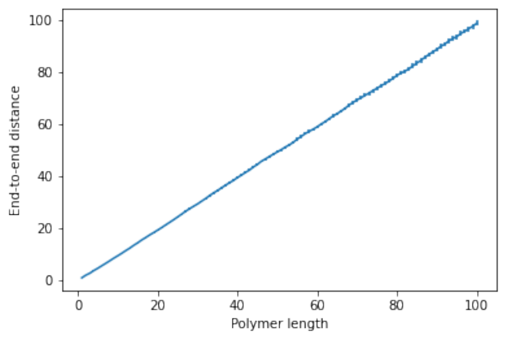


Figure 5: Average end-to-end distance as a function of polymer length with errorbars, calculated from a data-set of 10000 polymers modeled as FRW.

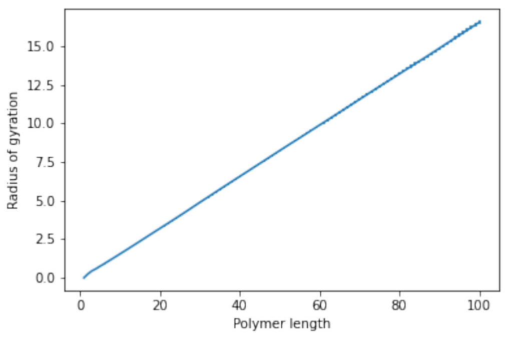


Figure 6: Average radius of gyration as a function of polymer length with errorbars, calculated from a data-set of 10000 polymers modeled as FRW.

These characteristic are known to scale according to equations (3) and (4), with $v = \frac{1}{2}$ in the case of modeling polymers as FRW, or in other words linear scaling with polymer length. This is clearly reflected in our results which show very good conformity to the expected behaviour.

3.4 Polymer generation

To determine the difference in generated polymers between the Rosenbluth method and PERM we calculated the ratio between the generated polymers with a length that was less than the target length and the polymers with a length of the target length. This ratio essentially shows how many polymers of a length less than the target length are generated for every polymer that has a length equal to the target length. To compare this ratio between the Rosenbluth method and PERM, for each target length the same amount of polymers with that target length were generated. The results are shown in figure 7, with a c_+ value of 2. It shows that for longer polymer target lengths more polymers have to be generated. At target lengths smaller than 60 the ratio is less than 1 for both the Rosenbluth method and PERM, and PERM has a slightly higher ratio than the Rosenbluth method. However, at polymer target length larger than 60 the ratio's start to diverge, while the ratio for PERM appears to increase linearly the ratio for the Rosenbluth method increases exponentially.

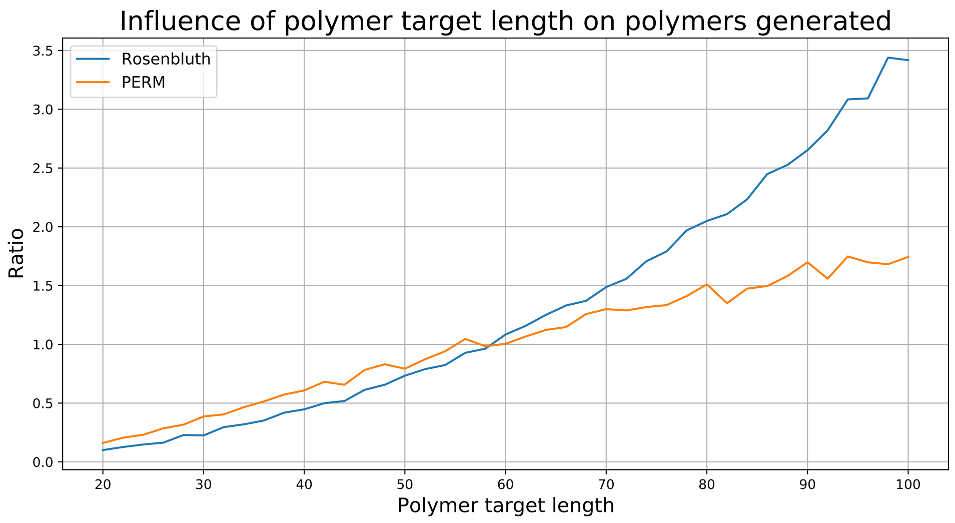


Figure 7: The ratio between the number of polymers generated that have the target length and the number of polymers generated that do not have the target length, for the polymers generated using the Rosenbluth method and the polymers generated using the PERM. For each polymer target length the same amount of polymers of that target length were generated. For the polymers generated using PERM the value of c_+ was set to 2.

3.5 Effect of c_+ on correlation and time spent

Understanding for the effect of changing the factor c_+ in the simulations was gotten by making two plots, one for estimating the relation between the correlation factor $\tilde{\mathcal{C}}$ and c_+, and one for estimating the relation between the time cost of generating polymers and the factor c_+, these are then used to plot a trade-off graph between correlation increase and speedup. The data were gathered by running the Rosenbluth method up to a length of 50 for 800 requested polymers; 800 polymers were needed to exhaust the possibility space of the large polymers. If significantly fewer polymers were chosen, there would be a possibility that we would generate unevenly more curled polymers than stretched polymers. This amount of polymers showed to obey the scaling laws. For the PERM we averaged 100 realisations of attempting to grow 10 polymers to a length of 90, this was done for each value of c_+.
The results for the relation between the correlation and c_+ in Figure 8 are not surprising. Due to the fact that less polymers are generated for higher values of c_+, the copied polymers have a greater influence on the overall correlation coefficient $\tilde{\mathcal{C}}$. The difference however does not seem to be that drastic since the difference between the standard Rosenbluth method and the pruned and enriched Rosenbluth method is only 12% for a c_+ value of 2. The correlation $\tilde{\mathcal{C}}$ seems to increase linearly with c_+ up to about $c_+ = 10$, after which it shows a sudden increase in correlation for $c_+ = 10.5$. This sudden jump is the effect of PERM breaking down for such high values, shown by the increased number of failures to find polymers of length L', with a failure rate of 95% for values greater than $c_+ = 10$.
c_+ does have a great influence on the amount of time the PERM needs to spend to find its polymer solutions as can be seen in Figure 9. The amount of time taken to find N representative polymers

up to target length L by PERM denoted by $T_{elapsed}$ seems to be inversely proportional to c_+ and is the result of the decreased amount of polymers generated, as seen in Figure 11. Thus, one can see that if you are willing to accept some extra correlation between polymers you can generate a set of representative polymers quicker for a higher value of c_+, as can be seen in Figure 10, a plot showing the trade-off between speedup and correlation increase.

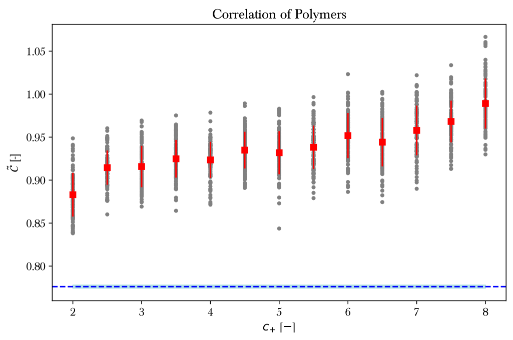


Figure 8: Plot showing the mean and standard deviation of the reduced correlation factor $\tilde{C}$ for 10 realisations of the standard Rosenbluth method with 800 polymers of length 50 using the dashed blue line and blue field respectively. The red boxes and lines denote the mean and standard deviation of $\tilde{C}$, respectively, for 100 implementations of the PERM method with 10 polymers to target length 90 with $c_+/c_- = 10$ fixed and c_+ given along the horizontal axis.

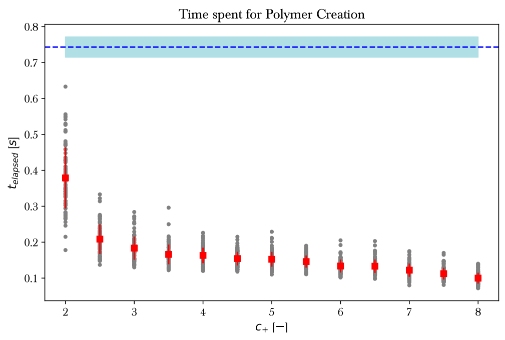


Figure 9: In the plot the mean and standard deviation of the time spent computing a realisation of 800 polymers of target length 50 are coloured blue for the standard method and 100 realisations of 10 polymers of length 90 are coloured red for the PERM method for a value of c_+

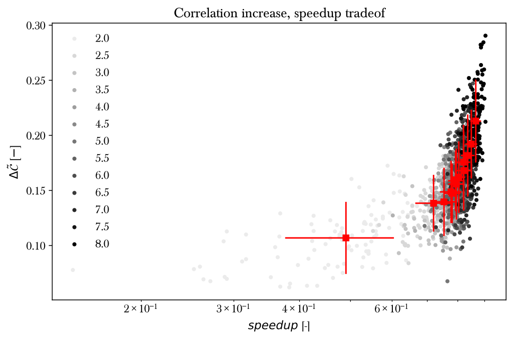


Figure 10: In the plot the correlation increase and speedup are plotted for all 100 realisations of 10 generated polymers per factor c_+

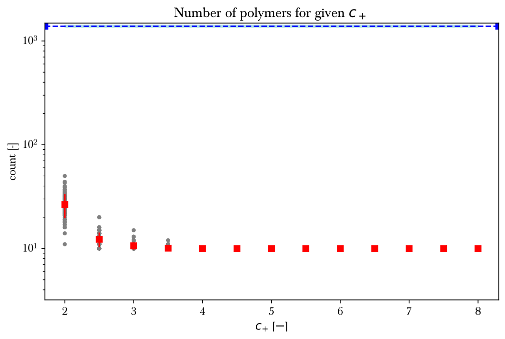


Figure 11: Plot shows the amount of polymers generated by Rosenbluth and PERM in blue and red, respectively. Plotted are the mean and standard deviation over 100 realisations of the generation of 10 polymers to length 90

4 Conclusion

To study the behaviour of the end-to-end distance and the radius of gyration in relation to polymer length, two sets of polymers were generated, one using the Rosenbluth method and one using the Pruned and Enriched Rosenbluth method (PERM). The PERM completed the creation of a representative set of polymers much quicker than the standard Rosenbluth method, as expected from the literature. For both data-sets a relation between the end-to-end distance and radius of gyration for different polymer lengths was sought. For both methods and data sets we found that the exponent of the scaling law, ν, was between 8% and 16% lower than was expected for two dimensional systems.

In contrast, the results of the end-to-end distance and radius of convergence for polymers modeled as free random walks showed great conformity to the expected linear scaling from literature. This is unsurprising since sampling these polymers is a lot more straightforward, and creation of a representative data-set is a far less computationally expensive, which allowed for sampling of 10000 polymers.

The comparison of the ratio of generated polymers between Rosenbluth and PERM showed that for larger polymer lengths more polymers are generated that have a smaller length for both Rosenbluth and PERM. However, as the polymer length increases PERM generates less polymers than Rosenbluth which indicates that PERM is less computationally expensive and that this difference becomes larger for larger polymer lengths.

Furthermore the introduced measure for overall correlation between polymers in a data set and the calculation thereof proved to be easily influenced by the amount of polymers in a data set. Instead of taking a weighted average it would probably be better to count individual entries in the matrix that exceed a threshold but this has not been tested in this report.

References

[1] M. N. Rosenbluth and A. W. Rosenbluth, “Monte carlo calculation of the average extension of molecular chains,” *The Journal of Chemical Physics*, vol. 23, pp. 356–359, Feb. 1955.

[2] P. Grassberger, “Pruned-enriched rosenbluth method: Simulations of θ polymers of chain length up to 1 000 000,” *Phys. Rev. E*, vol. 56, pp. 3682–3693, Sep 1997.

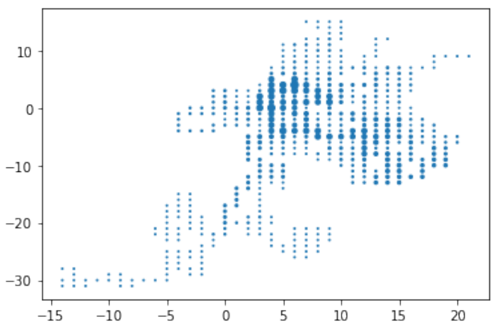
[3] M. Wimmer and A. Akhmerov, “Computational Physics ap3082,” 2022.

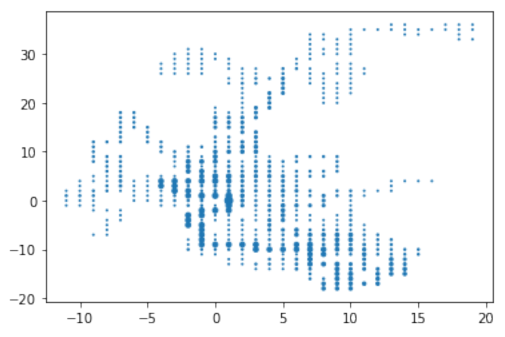
correlation

May 5, 2022

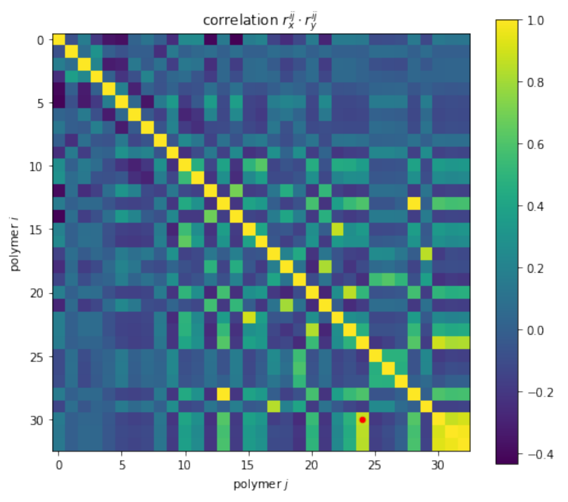
[ ]:
```
from polypymer.core_funcs import Polymer, Monomer, Dish
from polypymer.data_funcs import plot_dish, plot_polymer, grow_polymer, \
     generate_N_polymers, expect_observ, error_observ
import numpy as np
import matplotlib.pyplot as plt
```

[ ]:
```
dish = Dish((10,10),(5,5))
dish.PERM(10, 2, 100)

plot_dish(dish, stems=False)

dish.polymer_correlation(bouqet=True)

plot_dish(dish, bouqet=True, stems=False)

corr_matrix = dish.correlation
```





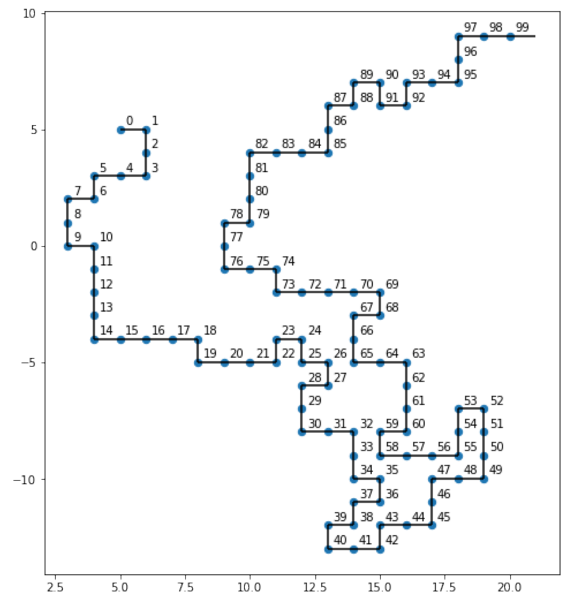
[ ]:
```
plt.imshow(corr_matrix, origin='upper')
plt.colorbar()
plt.xlabel(r"polymer $j$")
plt.ylabel(r"polymer $i$")
plt.title(r"correlation $r_{x}^{ij} \cdot r_{y}^{ij}$")
plt.scatter(24, 30, marker='.', s=80, color='red')
plt.gcf().set_size_inches(8,7)
plt.savefig('Figures/wow')
plt.show()
```

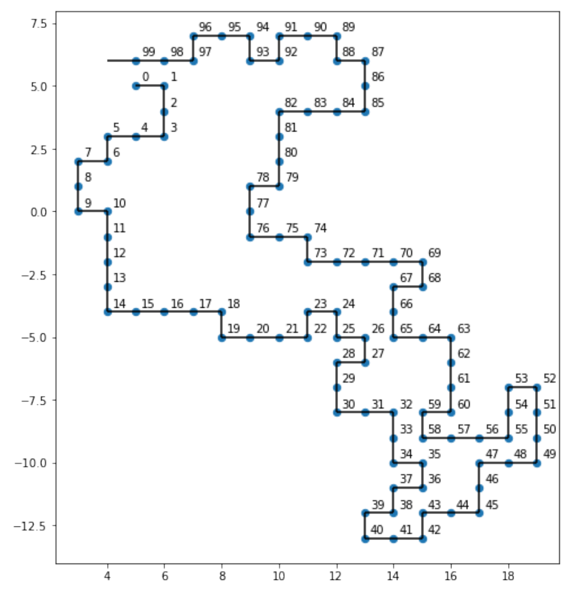


So according to our metric of correlation polymer 24 and polymer 30 should look alike

[ ]:
```
polymer_24 = dish.polymers[24]
polymer_30 = dish.polymers[30]
```

[ ]:
```
plot_polymer(polymer_24)
plot_polymer(polymer_30)
```





They do seem to like alike until node 87!

```
[ ]: def plot_polymer(polymer: object) -> None:

         x_ = np.asarray([])
         y_ = np.asarray([])

         xlines = np.asarray([])
         xlines_posx = np.asarray([])
         xlines_posy = np.asarray([])
         ylines = np.asarray([])
         ylines_posy = np.asarray([])
```

```
         ylines_posx = np.asarray([])

         (xmax, ymax) = polymer.dimensions
         cnt = 0

         for monomer in polymer:
             start = monomer.location
             end = monomer.end_location

             x_ = np.append(x_, start[0])
             y_ = np.append(y_, start[1])

             plt.text(x_[-1]+0.2, y_[-1]+0.2, str(cnt) )
             cnt += 1
             ang = monomer.angle
             if ang == 0 or ang == 2:
                 if ang == 0:
                     xlines_posx = np.append(xlines_posx, start[0])
                 else:
                     xlines_posx = np.append(xlines_posx, end[0])
                 xlines = np.append(xlines, 1)
                 xlines_posy = np.append(xlines_posy, start[1])
             if ang == 1 or ang == 3:
                 if ang == 1:
                     ylines_posy = np.append(ylines_posy, start[1])
                 else:
                     ylines_posy = np.append(ylines_posy, end[1])
                 ylines = np.append(ylines, 1)
                 ylines_posx = np.append(ylines_posx, start[0])


         plt.scatter(x_, y_, linestyle='None', marker='o', s=40)
         plt.vlines(ylines_posx, ylines_posy, ylines_posy+1, color='black')
         plt.hlines(xlines_posy, xlines_posx, xlines_posx+1, color='black')
         plt.gcf().set_size_inches(8,9)
         plt.show()
```