Nikita Dmitrieff, Clara Andiazabal, Parisa Kara, Samuel Kill and Peter Akinshin

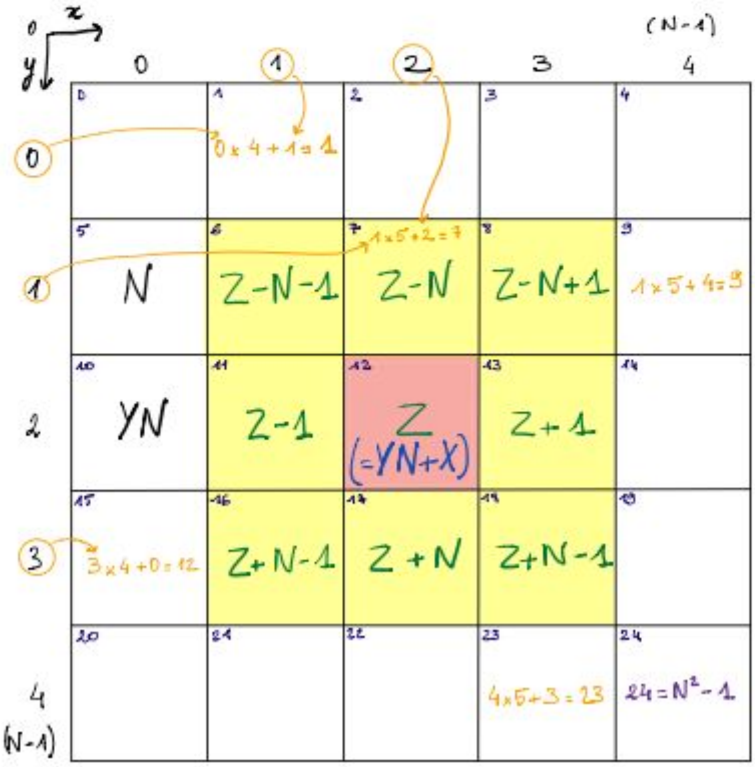
Polymer simulation supporting pdf

Table of Contents:

Order in which the functions/classes in the code are used:

1. class system()
    - def __init__()
    - def energy()
2. class bead()
3. def evolveOneStep()
4. def calcEnergyBead()
5. def whtCutoff()
6. def length()
7. def calcEnergy()
8. def calcTotAvEnergy()
9. def assignCell()
10. def assignCellBead()
11. def calcELJ()
12. def mapBead()
13. def findNeighbours()

Explanatory diagram for the findNeighbours() function:



In the code, nbCell = Z and side = N.
In this example, N = 5.
Z is the number of the cell that is considered (cell number 12 shown in red in this example).
The yellow cells are all the neighbouring cells for a given cell number Z.
In Orange, you can see examples of how we find the number of a cell in terms of its coordinates (Z=Y*N+X, in the code “aa * side + bb”)

The coordinates of the neighbouring cells in terms of Z help us create a list of all neighbouring cells in order to calculate the energy more efficiently (refer to the functions findNeighbours() and calcELJ() in the code)

Simulations:

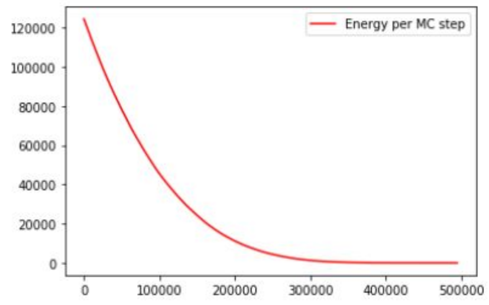
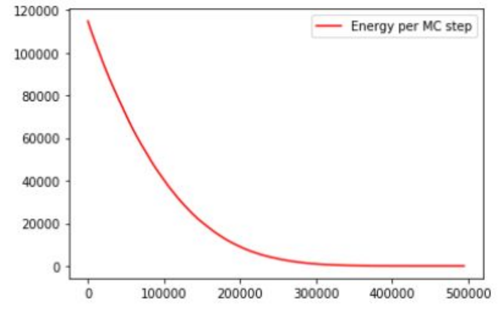
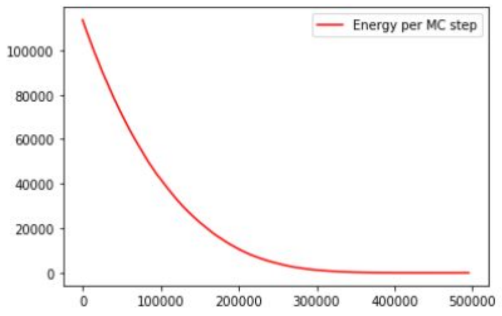
We ran a total of 330 simulations.

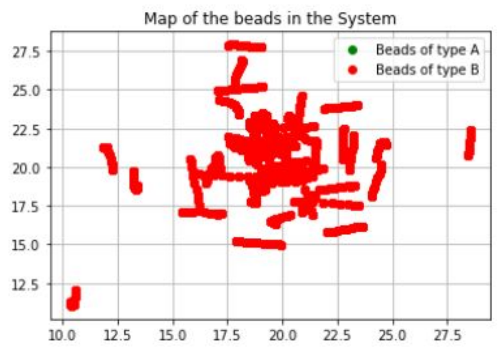
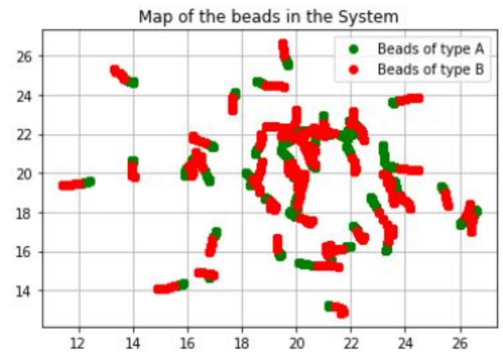
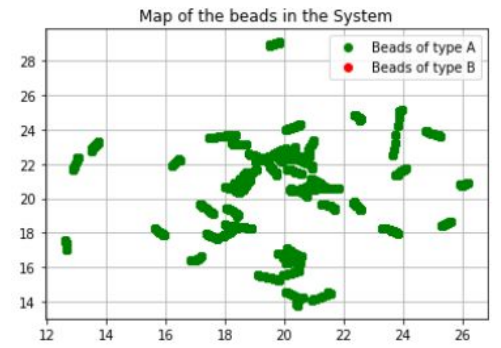
Since, the energy of the system during the first steps can be extremely high (depending on the inputs), we had to put a maximum value that we would display on our energy per MC step graph in order to display a coherent and interpretable graph. To do this, the simulation was run for 100 000 steps, the total energy recorded and the maximum value set to 1.1 * the recorded energy value. With this maximum value the code was ran for 500 000 steps, and the system's energy recorded every 5 000 steps (nSample = 5 000). This way, the graph is at a suitable scale, thus showing the relevant information.

To bypass this functionality, you can simply set "Max" to a very high value (10^99 for instance). This way, the energy of the system will always be inferior to "Max" thus always be recorded in the graph.

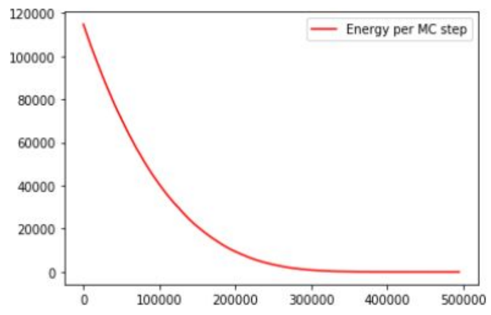
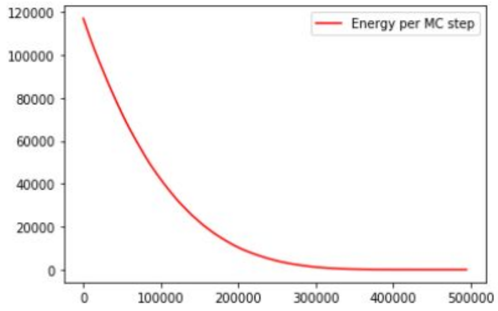
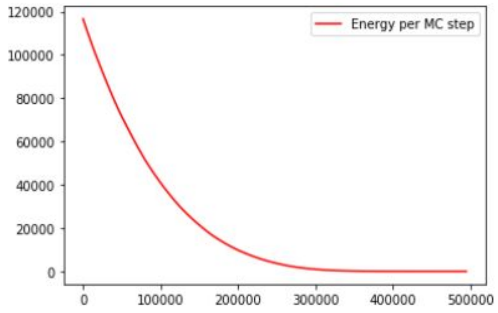
Here are a few screenshots of some interesting combinations of La and Lb:

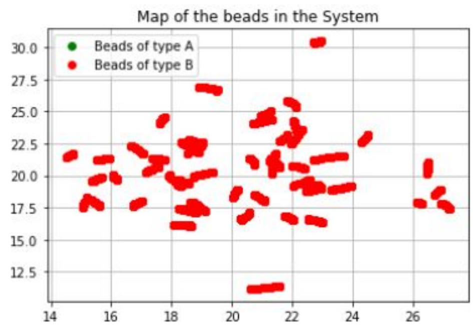
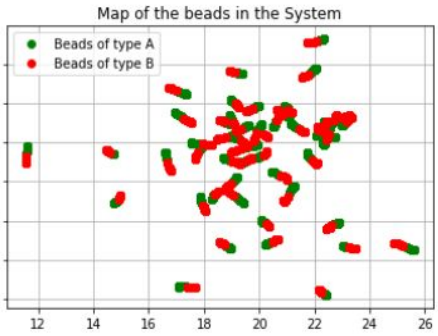
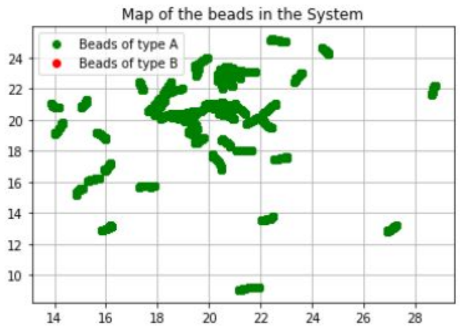
- For aa = 0.2:

La = 0 and Lb = 10	La = 5 and Lb = 5	La = 10 and Lb = 0
Total energy of the system : 13.999040382929783 Total average energy of the system: 0.027998080765859565	Total energy of the system: 5.816982818438822 Total average energy of the system: 0.011633965636877644	Total energy of the system : 2.109935815150023 Total average energy of the system: 0.004219871630300047
		

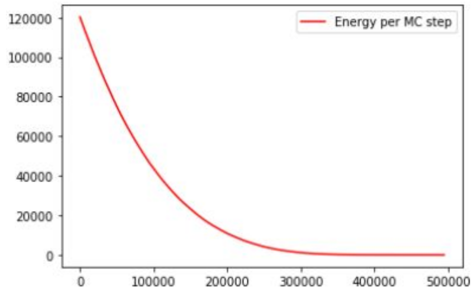
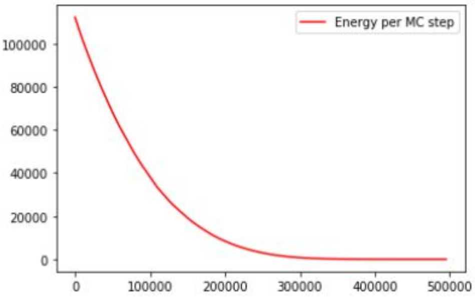
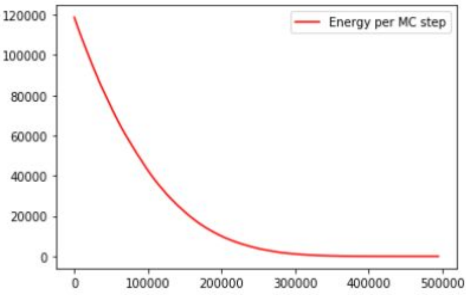
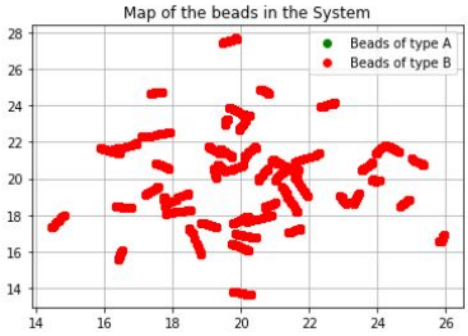
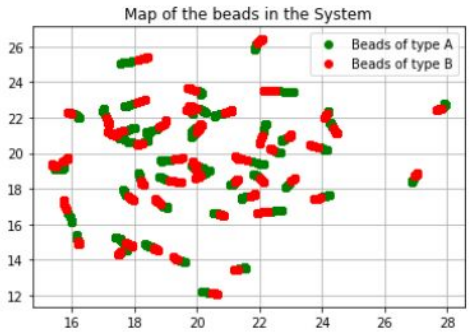
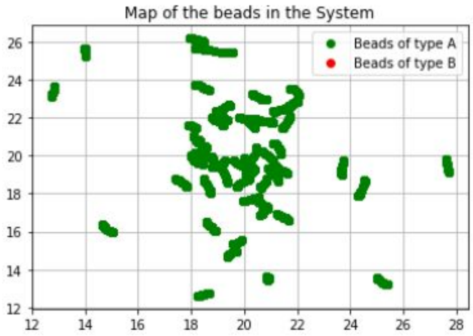
		

- For aa = 1.0:

La = 0 and Lb = 10	La = 5 and Lb = 5	La = 10 and Lb = 0
Total energy of the system : 1.4882445352047293 Total average energy of the system: 0.0029764890704094588	Total energy of the system: 1.5626907069691458 Total average energy of the system: 0.0031253814139382916	Total energy of the system : 1.6069125717379258 Total average energy of the system: 0.0032138251434758514
		

- For aa = 2.0:

La = 0 and Lb = 10	La = 5 and Lb = 5	La = 10 and Lb = 0
Total energy of the system : 1.4291124880463089 Total average energy of the system: 0.0028582249760926177	Total energy of the system: 1.5889961328371118 Total average energy of the system: 0.0031779922656742236	Total energy of the system : 1.7544165261620905 Total average energy of the system: 0.0035088330523241813
		
		

Heatmap:

Please ensure the csv file provided is in the correct section of your computer (in the same folder as the code). This is so that when the code is run, the csv file is found and the heatmap is generated.

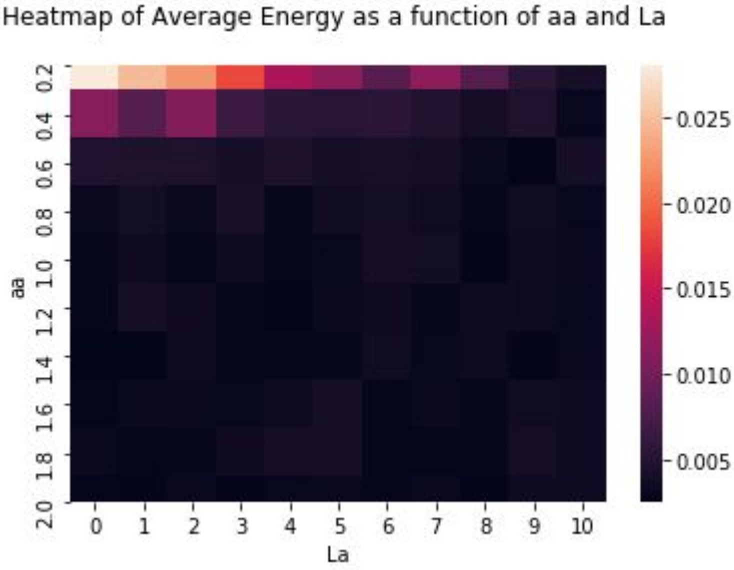
If not, below are screenshots of the heatmaps:

N/B:

We chose to graph 2 heatmaps, one with the three "eps" multiplied by the constant k (giving rise to the real values of energy) and one without in order to accentuate the weighting of ELJ in the total energy. This enabled us to see in much greater detail the variation in average energy as a function aa and bb.

We studied the first heatmap (that with the real values of energy) and later decided to focus on the second (that without the constant k considered) as it unveiled some much more interesting trends for us to analyse.

The first heatmap- that with the constant k considered:

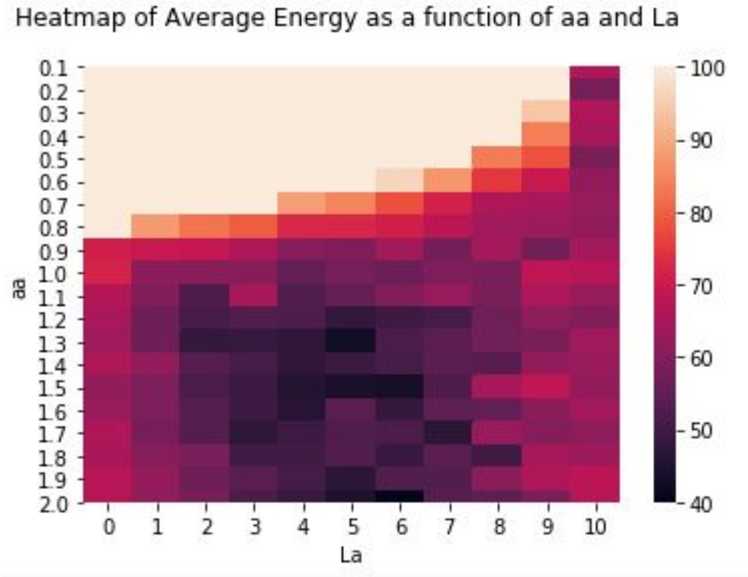


Analysis of the first heatmap:

The majority of the graph is black/purple which doesn't allow for many trends to be identified, hence for the second heatmap. The reason for this is due to the fact that most of the energy values are within the 0-0.075 range.

To then highlight the variations in energy of this region, we decided to remove the constant k responsible for these low values. Since k is a constant, removing it would not have affected the variance in the values.

The second heatmap- that with the constant k <u>not</u> considered:



To create this heatmap, we ran 220 simulations and recorded the value of the average energy for each of the combinations of aa (from 0.1 to 2) and La (from 0 to 10).

As explained by the comments in the code, we decided to plot the heatmap as a function of aa and La (instead of bb), as this would insure that La and Lb were integers (since they are respectively the numbers of beads A and B in the polymer).

For the colorbar, we chose a minimum value of 40 and a maximum of 100, since most of the energy values fall within 40 and 100.. This way, a range of colors corresponding to a range of average energies could be seen.

The light pink part in the top left corner is due to energy values astronomically higher than 100 and the parameters associated with them.

The colorbar would have been better shown as a logarithmic scale however, this then distorted the colours making the graph illegible.

Analysis of the second heatmap:

The reason for the top left corner being above 100 is due to the value of aa being very low. This drastically increases the rcutoff of type B beads, thus increasing the number of interactions. Furthermore, the top left has a higher number of type B beads relative to type A beads, contributing to the large values of total energy.

The simulations running with 10 type A beads all have energies in same range (around 60-70), which is not surprising as aa does not make any difference for type A beads (unless there are some type B beads, having an impact on sigma_AB), so the simulations should be equivalent - the results show this.

The average energy of La = 0 decreases as aa increases, this is also expected as the increase in aa correlates to rcutoff of type B beads decreasing (and that of type A + B interactions too).

These are only a few of the interesting trends unveiled by this heatmap. Some of them are hidden in the top most left corner due to the collosal values of energy associated with them.

Both heatmaps together allow the trends across the entire graph to be identified. The first heatmap shows the upper left corner in detail and the second the rest.