PH456 Lab 1 : Random Numbers

Sourabh Choudahary

March 27, 2022

The code is available on github via this link. Code for **Task 1** is in lab-1.ipynb and for **Task 2 to 4** is in lab-1-task2-4.py.

1 Generating random numbers from different initial seeds

Two sets of random numbers were generated, first using python built-in module **random** and installed library **numpy**.

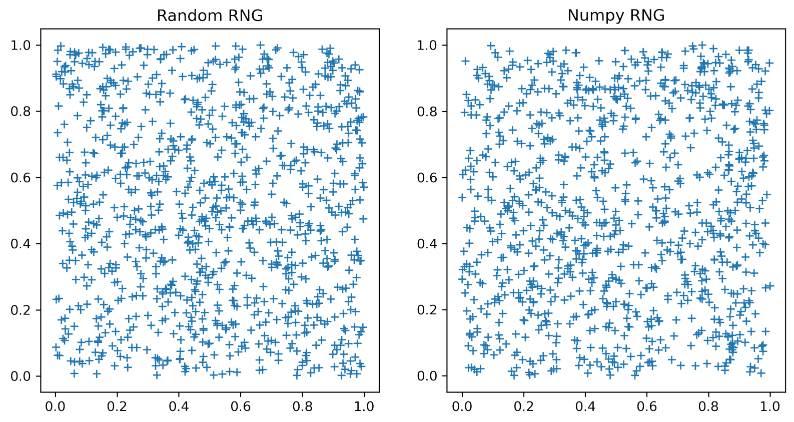

Figure 1: These figures show 1000 random numbers plotted in 2D. The figure on the left shows the random numbers generated using the python built module **random**. The figure on the right shows the numbers generated using the library **numpy**.

1.1 Testing for unifomity using χ^2 test:

The uniformity for both random number generators (RGNs) is visualised in Figure 1 used and was tested using the χ^2 test. Following equations show the results for the python built-in module **random** and for the numbers generated using the library **numpy** respectively.

$$\chi_1^2 = 83, p = 0.88 \tag{1}$$

$$\chi_2^2 = 94, p = 0.62 \tag{2}$$

To do the χ^2 test the **frequencies** function was written to generate the frequency distribution for the random number using given array of numbers and number of bins.

Test was done with random numbers N = 1000, Bins = 100. From the results we can see in both cases $\chi^2 < 100$. This shows that the generated random numbers are uniform and pass this test. The p values show the probability of obtaining $\chi^2 \geq$ current value and still pass the test.

1.2 Auto-correlation :

Using the statsmodels library the autocorrelation function for 100 Random numbers was plotted as shown below:

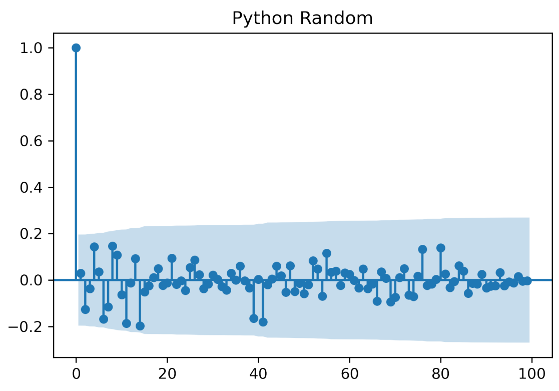

Figure 2: Figure on left shows the auto-correlation for 100 Random Numbers generated using python built-in module **random**. The scale works measures the correlation on a scale between 1 and -1. With 1 indicating a 100 % positive correlation and -1 indicating a negative correlation. The shaded portion represents the confidence interval determined using Bartlett's formula, plotted here with a 95 % confidence interval. Points in the shaded region show that there is no significant correlation.

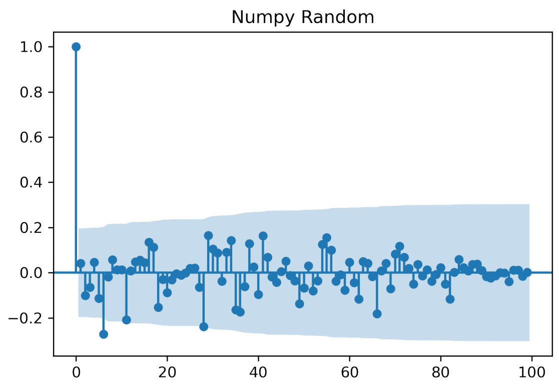

Figure 3: Figure on left shows the auto-correlation for 100 Random Numbers generated using **numpy** library. With a similar interpretaion as in the previous figure, this too shows that there is no significant correlation between the numbers generated.

2 Simulating a Partitioned Box:

Running the **lab-1-task2-4.py** will start the animation of the particle simulator. The simulator was built using the code found on this website.
Comments have been added with guidlines to the code to switch between the different RNGs for inital conditions of the simulations.

3 Partitioned box simulations:

3.1 Using built-in module random for inital conditions:

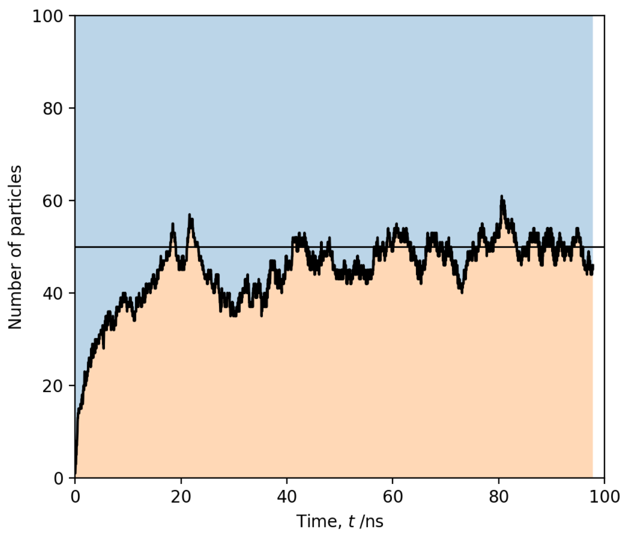

Figure 4: Figure on left shows simulation for 100 particles with inital positions generated using **random** module, the velocities and orientations were also generated using the same. Particles are *on average* equally distributed in the two sides as expected.

3.2 Using numpy random numbers for inital conditions:

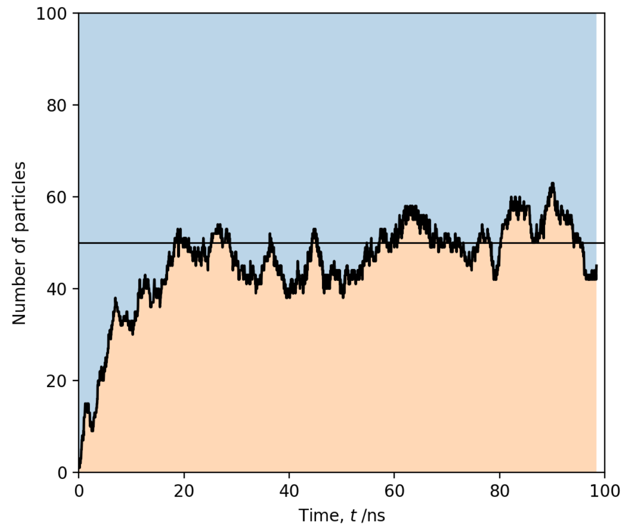

Figure 5: Figure on left shows simulation for 100 particles with inital positions generated using **numpy** library, the velocities and orientations were also generated using the same. Results are similar as seen before.

4 75:25 Probabilty for the partitioned box :

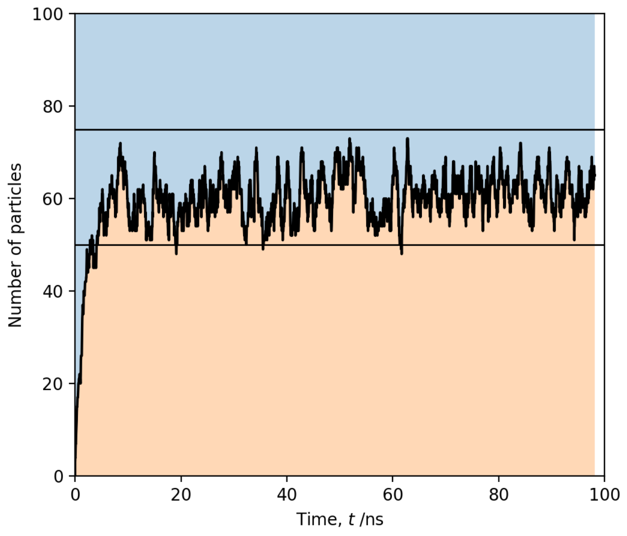

Figure 6: The intention of this simulation was to have 75 % of the particles on one side of the box. Attempted to achieve this by manipulating the reaction that particle has on hitting the middle wall, but it did not work as intended.

5 Link

github repository :