Homework 4 Write Up

Jalyn-Rose Clark

November 8, 2024

1 Random Numbers

The overall usage of this program is to initialize and call random numbers from a random number generator.

1.1 1,000 Random Numbers Evenly Distributed From 0-1

1000 random numbers evenly distributed between 0 and 1 were generated using the python package **random.random**. The uniform random numbers were then plotted using a histogram.

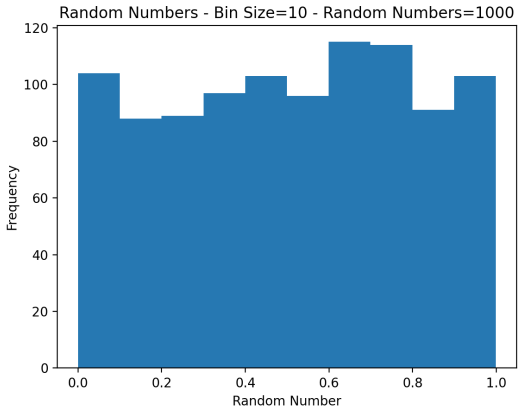


Figure 1: Bin Size: 10

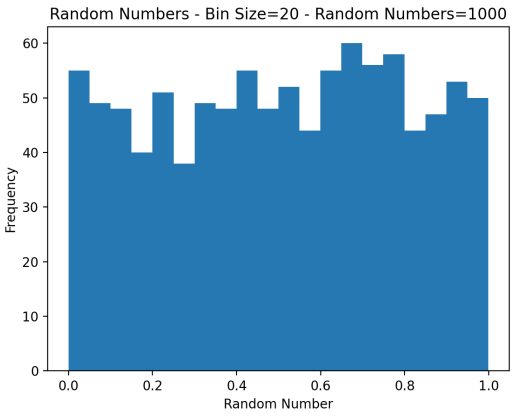


Figure 2: Bin Size: 20

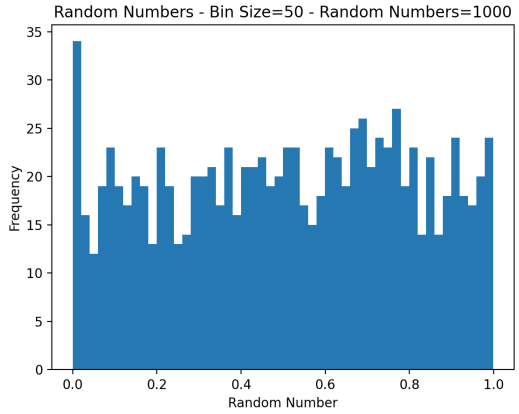


Figure 3: Bin Size: 50

Figure 4: Bin Size: 100

1.2 1,000,000 Random Numbers Evenly Distributed From 0-1

1000000 random numbers evenly distributed between 0 and 1 were generated using the python package **random.random**. The uniform random numbers were then plotted using a histogram.

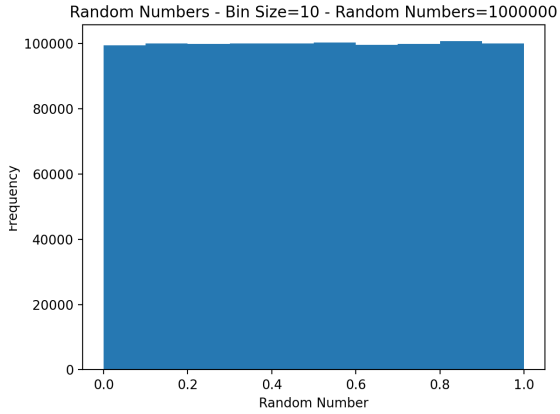


Figure 5: Bin Size: 10

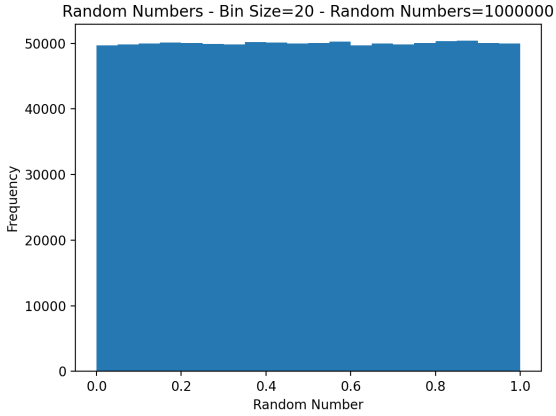


Figure 6: Bin Size: 20

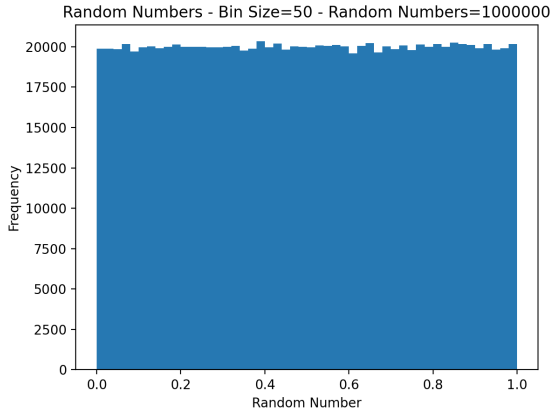


Figure 7: Bin Size: 50

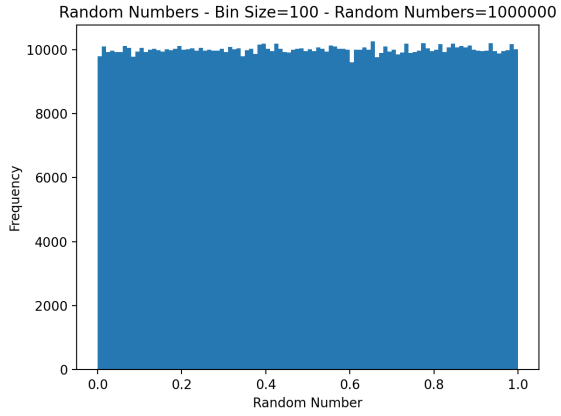


Figure 8: Bin Size: 100

1.3 Random Numbers With a Gaussian Distribution

The same random number generator was used along with the Box-Mueller Method [1]. With this method 2 uniformly randomly distributed sets of numbers between 0 and 1 are distributed, U_1 and

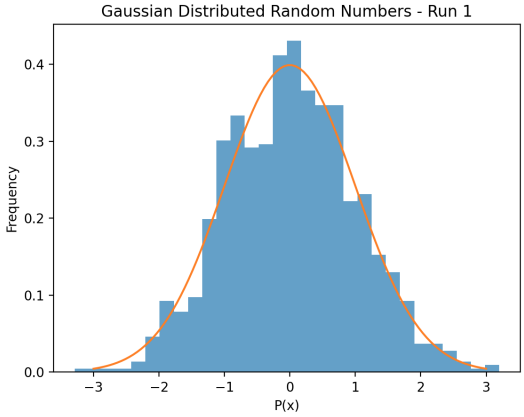
U_2. These randomly distributed numbers can be transformed into Cartesian coordinates representing the Gaussian distribution.

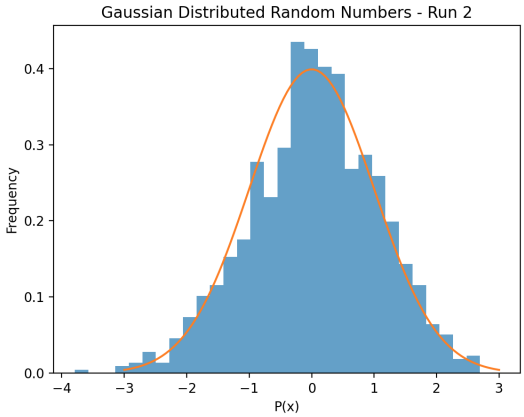
$$X = \sqrt{-2ln(U_2)}cos(2\pi U_1) \tag{1}$$

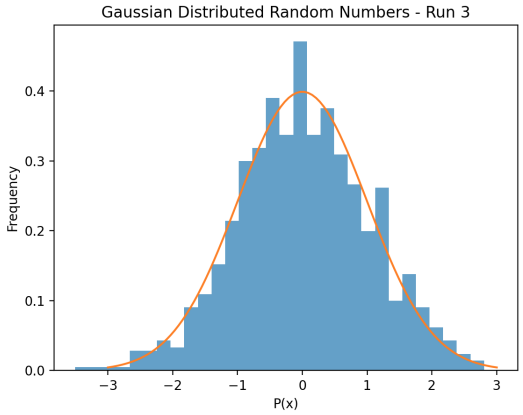
$$Y = \sqrt{-2ln(U_2)}sin(2\pi U_1) \tag{2}$$

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{3}$$

3 different trials were performed using different seed values to verify the algorithm along with overlaying a Gaussian onto the plots.







2 2D Random Walk

This program simulates a random walk in 2D on a discrete square lattice.

2.1 Mean Displacement and Average of the Square Position

10^4 random walks were averaged for 100 iterations where if the random number was between 0 and 0.5 the walk moved +1 in x and y-direction and if the random number was between 0.5 and 1 the walk moved -1 in x and y-direction. Each random walk was normalized by the total number of random walks. $\langle x_n \rangle$, the mean displacement after n steps was plotted for the averaged walks.

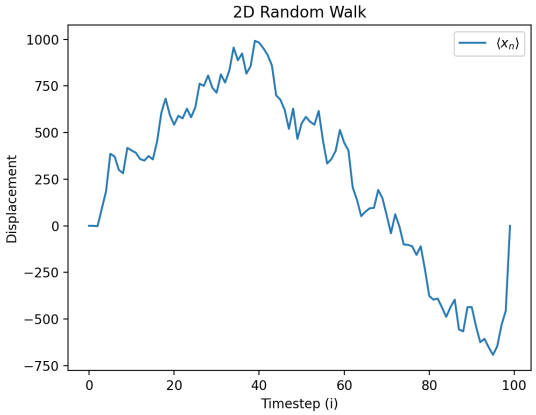


Figure 9: $\langle x_n \rangle$ for a 2D Random Walk

$\langle (x_n)^2 \rangle$, the average of the square position at step n was plotted for the averaged walks.

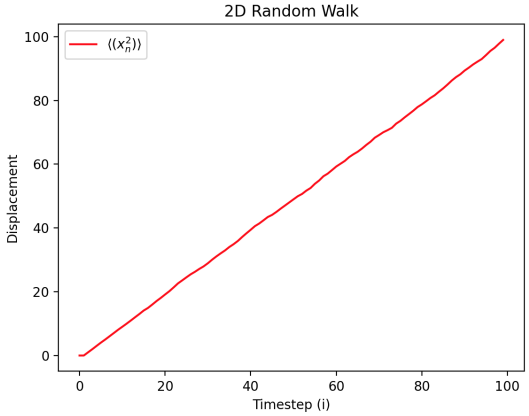


Figure 10: $\langle (x_n)^2 \rangle$ for a 2D Random Walk

2.2 Mean Square Distance

To show that the motion is diffusive, the mean square distance from the starting distance, $\langle r^2 \rangle$ was plotted as a function of the time step.

$$\langle r^2 \rangle = \langle (x_n)^2 \rangle + \langle (y_n)^2 \rangle \tag{4}$$

Using the **numpy.polyfit** package in python a linear polynomial line was fit to the $\langle r^2 \rangle$ vs time-step line to find the diffusion constant.

$$D = \text{slope}/2 \tag{5}$$

$$D = 2.015 \tag{6}$$

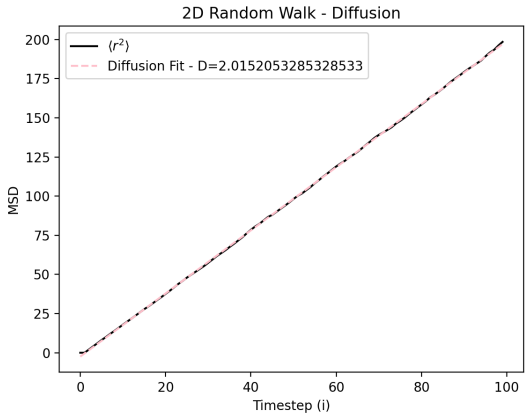


Figure 11: $\langle r^2 \rangle$ for a 2D Random Walk

3 Diffusion Equation

3.1 Analytical Spatial Value of 1D Normal Distribution

The spatial expectation value of the 1D normal distribution is given by:

$$\langle x(t)^2 \rangle = \int_{-\infty}^{\infty} x^2 \rho(x,t) dx \tag{7}$$

And the 1D normal distribution is given by:

$$\rho(x,t) = \frac{1}{\sqrt{2\pi\sigma(t)^2}} \exp\left(-\frac{x^2}{2\sigma(t)^2} \right) \tag{8}$$

Solving the inegral gives us:

$$\langle x(t)^2 \rangle = \int_{-\infty}^{\infty} x^2 \frac{1}{\sqrt{2\pi\sigma(t)^2}} \exp\left(-\frac{x^2}{2\sigma(t)^2} \right) dx \tag{9}$$

$$= \frac{1}{\sqrt{2\pi\sigma(t)^2}} \int_{-\infty}^{\infty} x^2 \exp\left(-\frac{x^2}{2\sigma(t)^2} \right) dx \tag{10}$$

$$= \frac{1}{\sqrt{2\pi\sigma(t)^2}} \sigma(t)^2 \sqrt{2\pi\sigma(t)^2} \tag{11}$$

$$= \sigma(t)^2 \tag{12}$$

3.2 Numerical Solution to Diffusion Equation Using Finite Difference Form

This program solves the 1D diffusion equation using the finite difference form with a diffusion constant D=2. The initial density profile peaks around 1 over a few grid sites. The later times were plotted to confirm that they respond to a normal distribution with $\sigma(t) = \sqrt{2Dt}$.

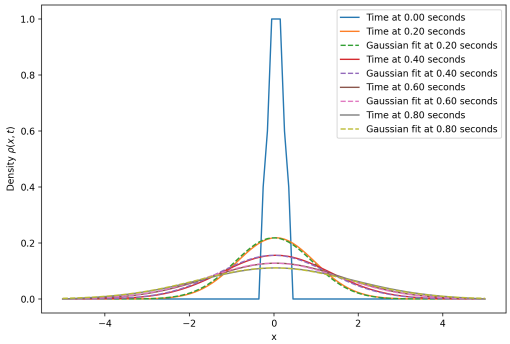


Figure 12: 1D diffusion equation solved using the finite difference form.

4 Mixing of 2 Gases

This program simulates the mixing of two gases in 2D in a rectangular enclosure.

4.1 Gas Particles A and B Mixing Randomly

A 60x40 grid was used with the left third of the grid containing gas A and the right third containing gas B and the middle being empty. The gas particle could move to a random location on the grid either left, right, up, or down and if the position is occupied it moves to another site. This was done over 100 trials using the python packages **random.randint** and **random.choice**.

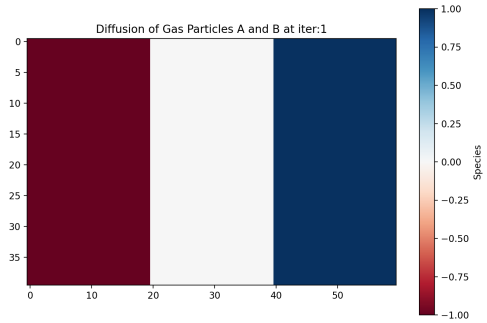


Figure 13: Diffusion of gas particle A and gas particle B at iteration 1.

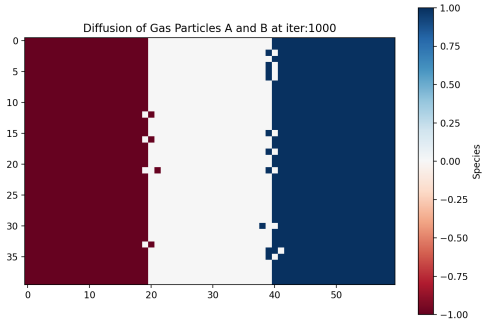


Figure 14: Diffusion of gas particle A and gas particle B at iteration 1,000.

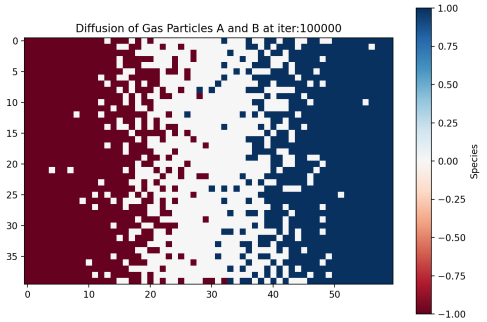


Figure 15: Diffusion of gas particle A and gas particle B at iteration 100,000.

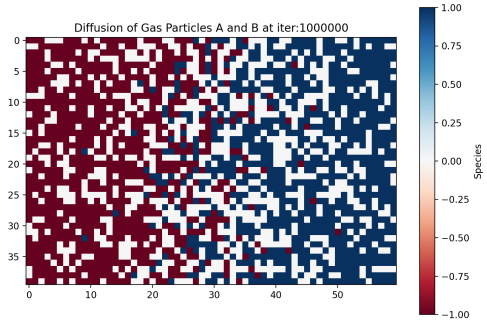


Figure 16: Diffusion of gas particle A and gas particle B at iteration 1,000,000.

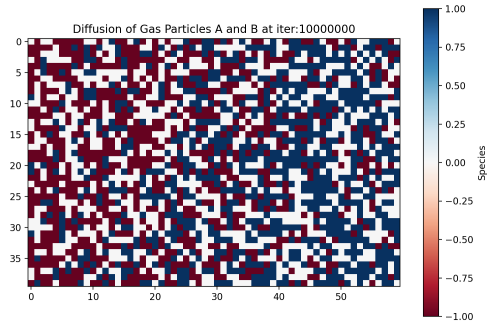


Figure 17: Diffusion of gas particle A and gas particle B at iteration 10,000,000

4.2 Linear Population Densities

The linear population densities $n_A(x)$ and $n_B(x)$ after specific time intervals were plotted.

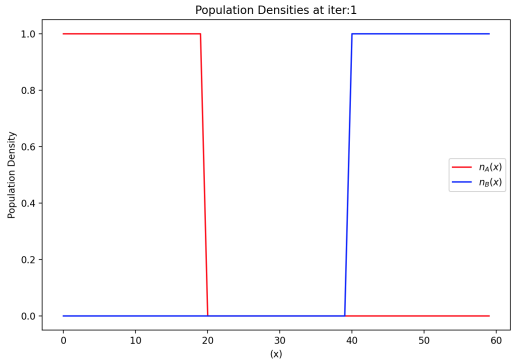


Figure 18: Linear population densities of gas particle A and gas particle B at iteration 1.

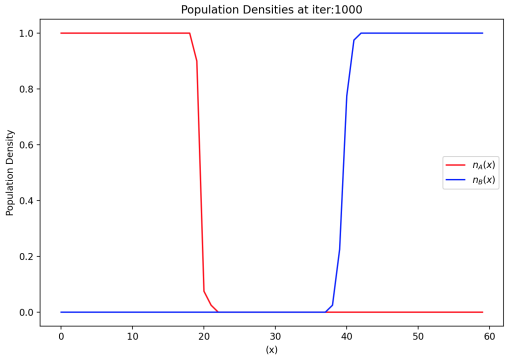


Figure 19: Linear population densities of gas particle A and gas particle B at iteration 1,000.

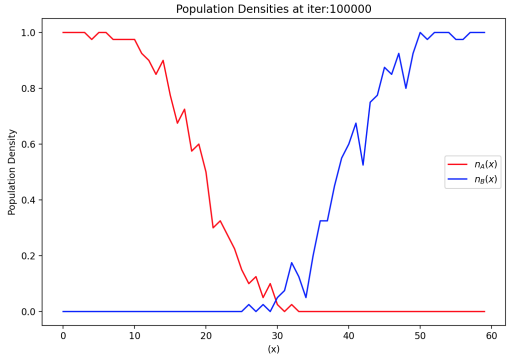


Figure 20: Linear population densities of gas particle A and gas particle B at iteration 100,000.

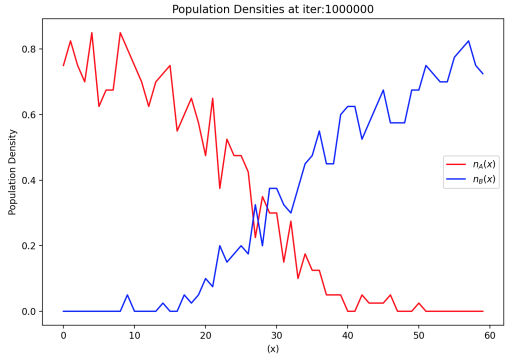


Figure 21: Linear population densities of gas particle A and gas particle B at iteration 1,000,000.

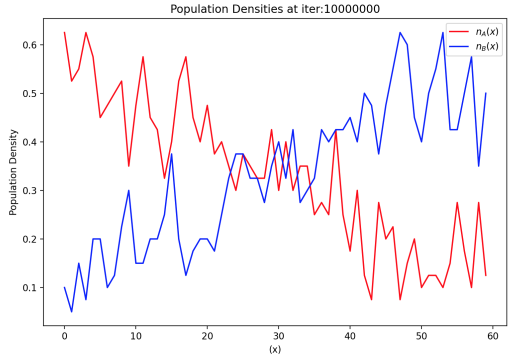


Figure 22: Linear population densities of gas particle A and gas particle B at iteration 10,000,000.

4.3 Average Gas Densities

The average gas densities over 100 trials were then averaged for accuracy.

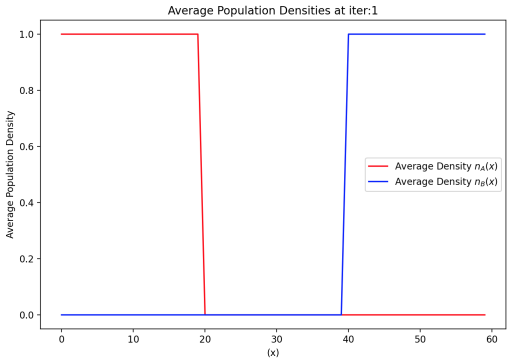


Figure 23: Averaged linear population densities of gas particle A and gas particle B at iteration 1.

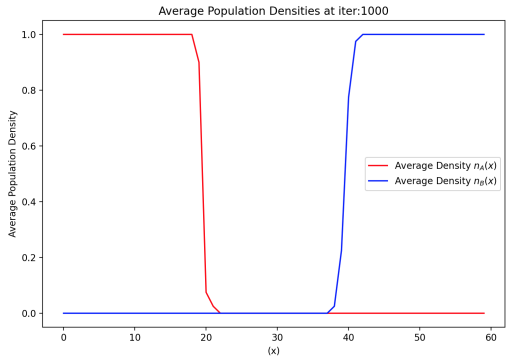


Figure 24: Averaged linear population densities of gas particle A and gas particle B at iteration 1,000.

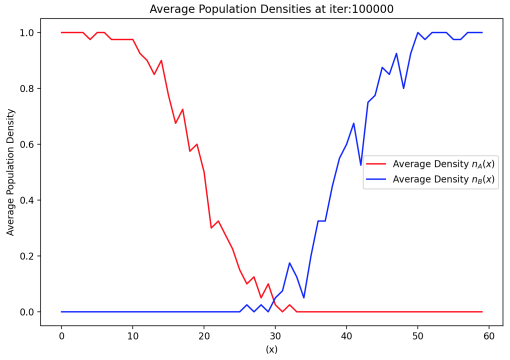


Figure 25: Averaged linear population densities of gas particle A and gas particle B at iteration 100,000.

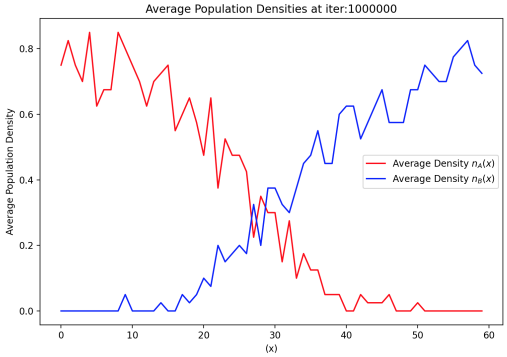


Figure 26: Averaged linear population densities of gas particle A and gas particle B at iteration 1,000,000.

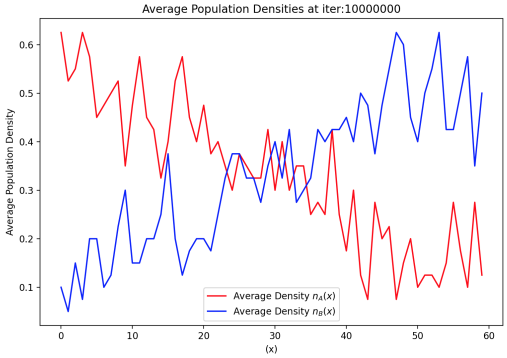


Figure 27: Averaged linear population densities of gas particle A and gas particle B at iteration 10,000,000.

References

[1] Cupcake Physics. *The Box-Muller Algorithm*. Accessed: 2024-11-07. 2015. URL: https://cupcakephysics.com/computational%20physics/2015/05/10/the-box-muller-algorithm.html.