Problem 1

Brute Force Simulation

We will consider a simple two-state (A and B) stochastic process, characterized by the dynamics:

$$A \xrightarrow{\omega_{AB}} B;$$

$$B \xrightarrow{\omega_{BA}} A,$$

with transition rates $\omega_{AB} = 0.5$ and $\omega_{BA} = 1.5$.

At first, we are going to simulate it using a simple brute force algorithm for a time $T = 100$ time units (t.u.), with a time-step $\mathrm{d}t = 10^{-3}$.

First, we import the necessary packages:

Now we write two functions for:

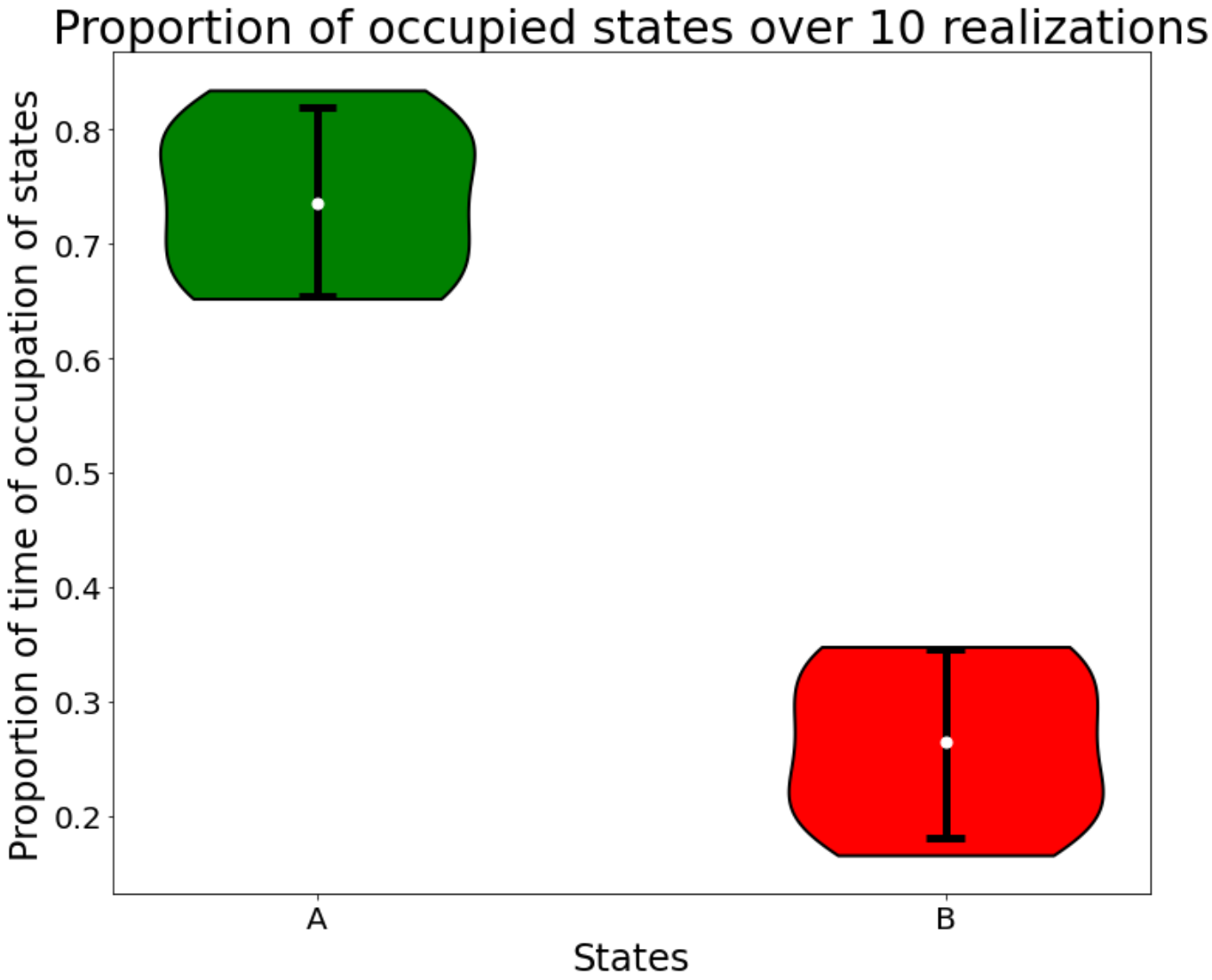
1. Generating an array with the evolution of the system;
2. Running multiple realizations while saving the statistics of state occupation, over multiple realizations of the process.

Let's now generate a few realizations and look at the time evolution:

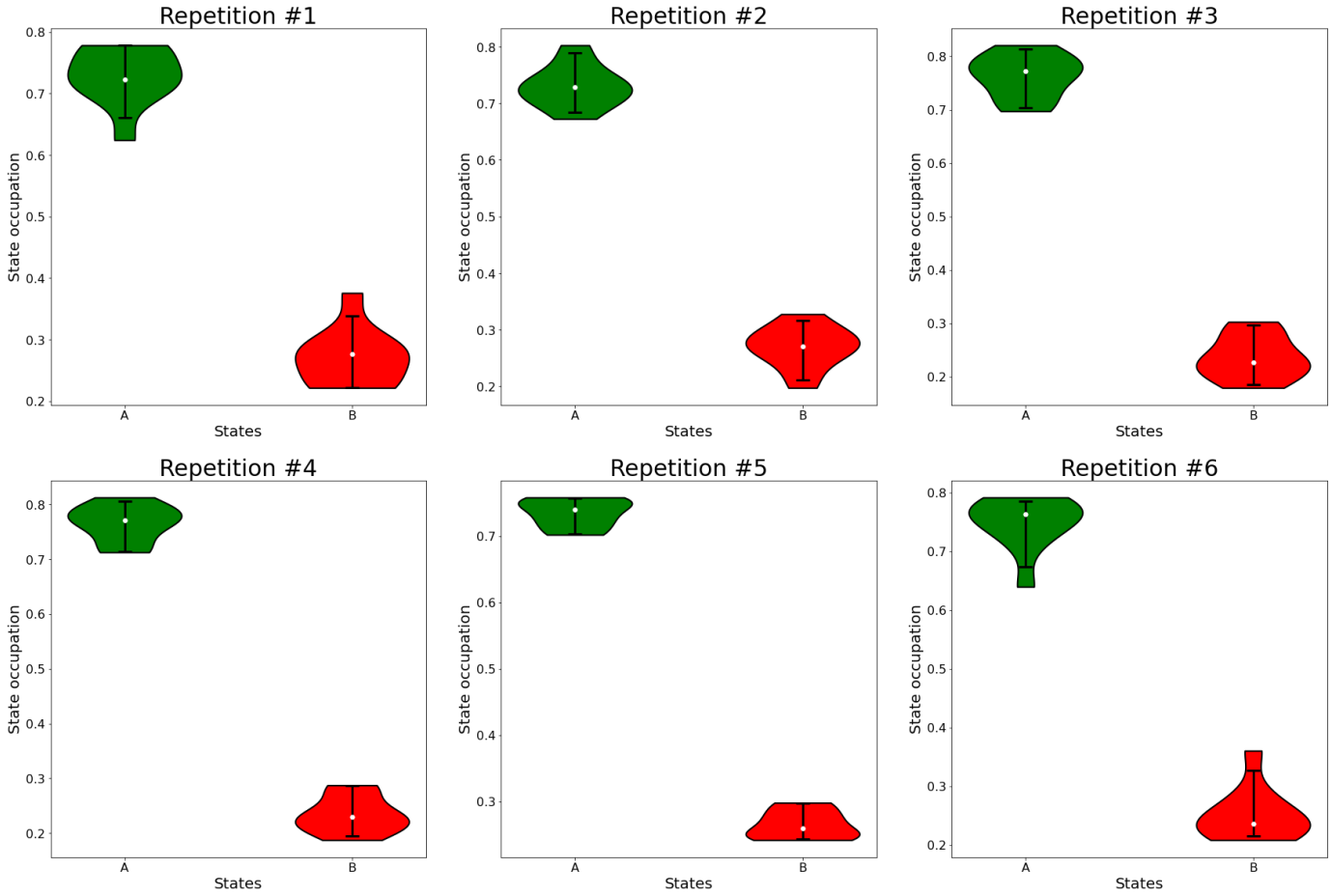
We can clearly see the behavior of the system, but its hard to do a quantitative analysis of the

results looking at the data like this. Let's take the statistics of how many of the points are on each state, considering all figures. That is, let's look at the state occupation statistics.

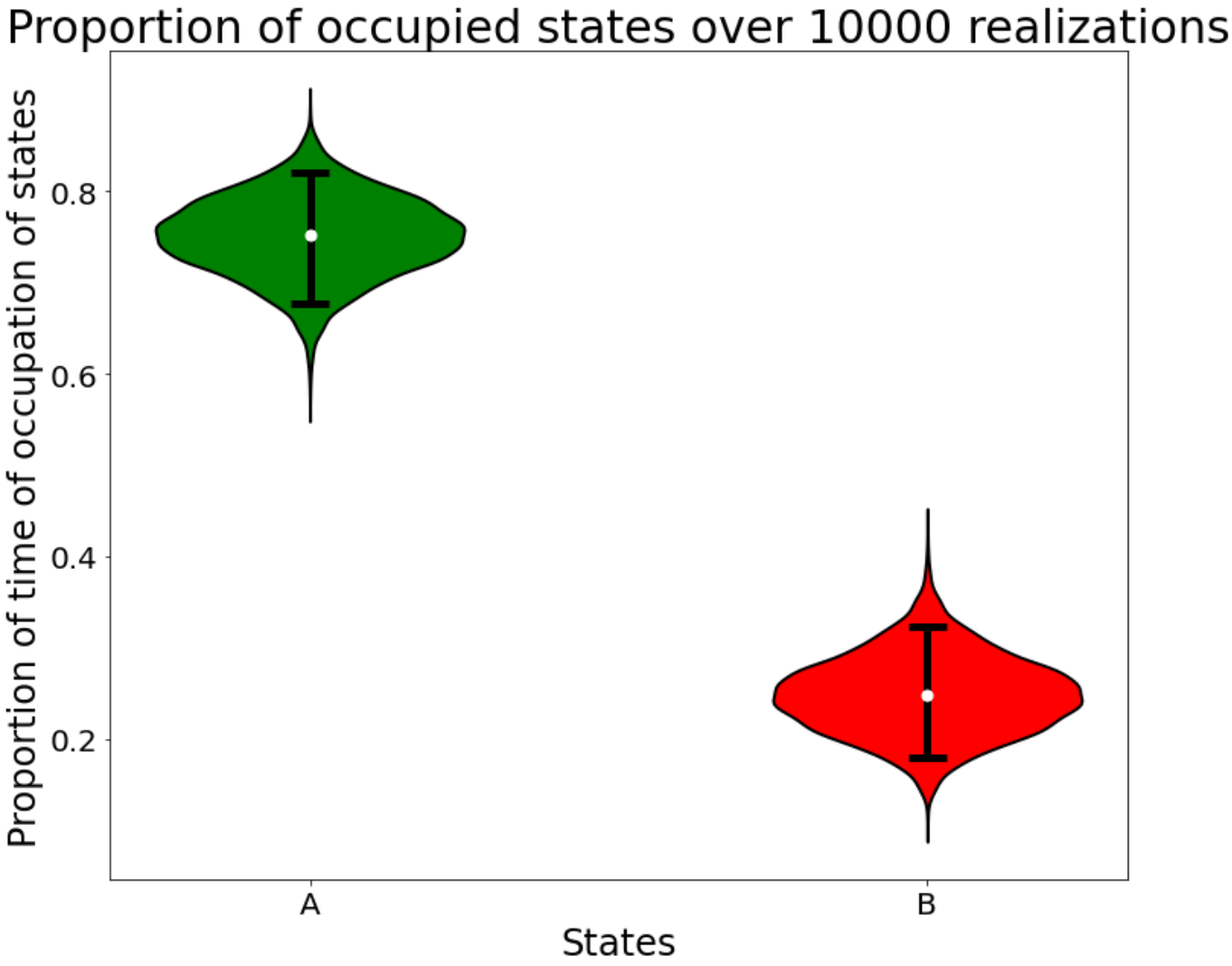
As $\omega_{BA} = 3\omega_{BA}$, we would expect the system to spend 75% of the time on state A, and the rest on B. Let's look at the actual results.



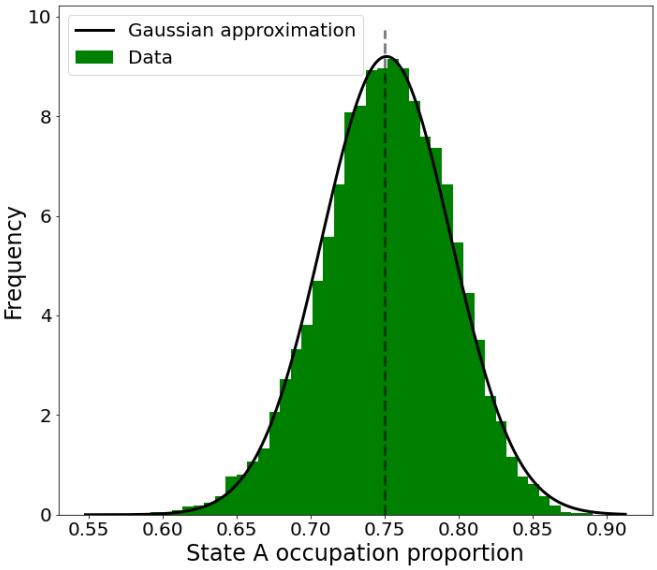
We see that the resulting distributions of state occupations are close to the expected values, but the distributions are rather uneven. Notice that the white points indicate the average, and the bars indicate the condifence interval from 5 to 95 percentile. Lets run this a few more times and look at how it varies.

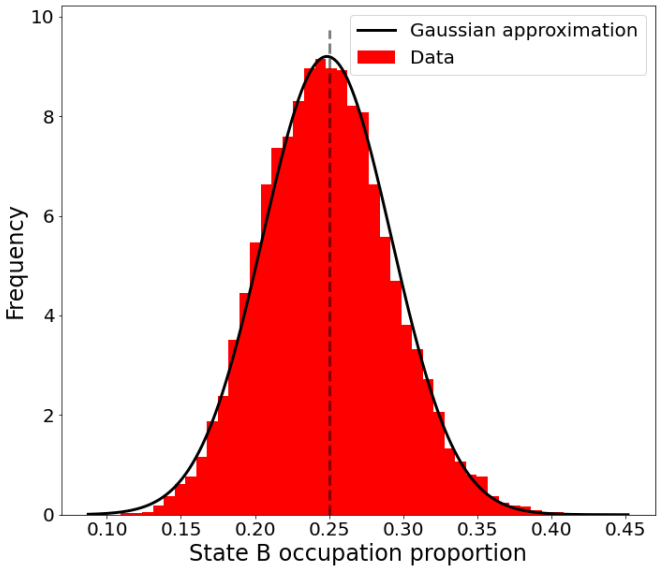


There is a noticeable variation over different repetitions, maybe we can get less variating results by increasing the number of realization of the process we take into account. Let's consider 10^4 realizations:



These results look better, and we can even look at the distribution of occupation for each state, over all realizations:





We can see that the distribution of occupation of each state over multiple realizations can be nicely approximated by a normal distribution, and their centers are close to the expected values. Let's look at the actual values:

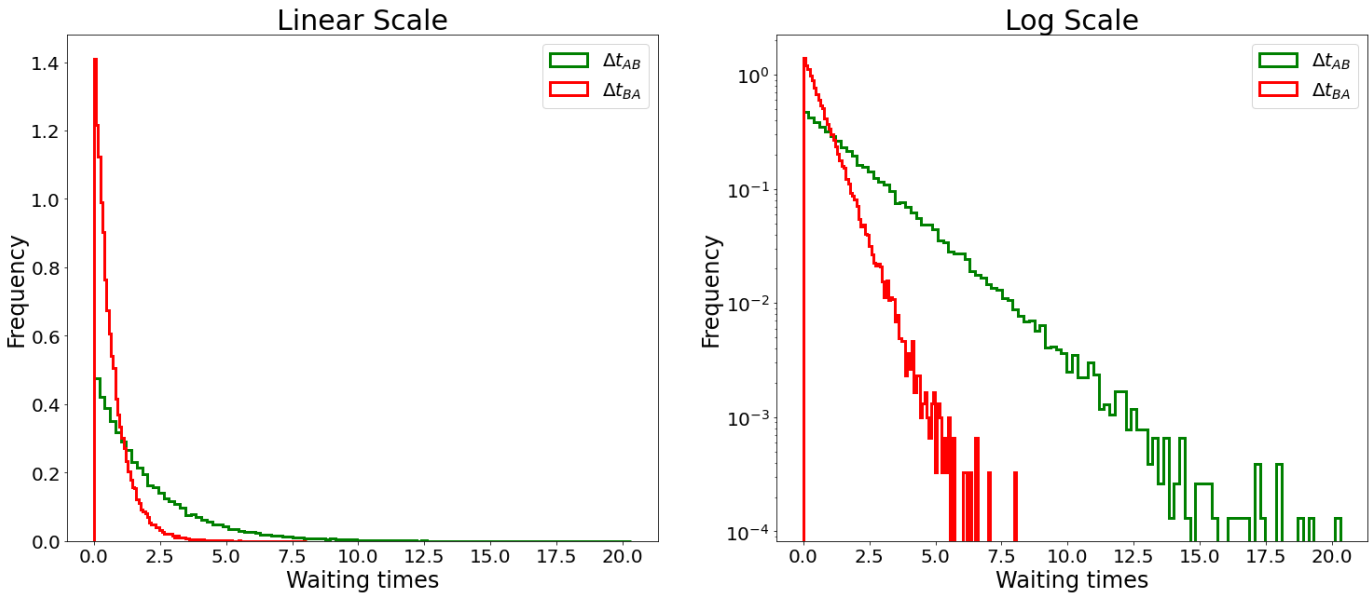
```
State A occupation distribution:
        Average: 0.7513724030000001
        Standard Deviation: 0.04336470490428351
State B occupation distribution:
```

```
Average: 0.248627597
Standard Deviation: 0.04336470490428351
```

Waiting time statistics

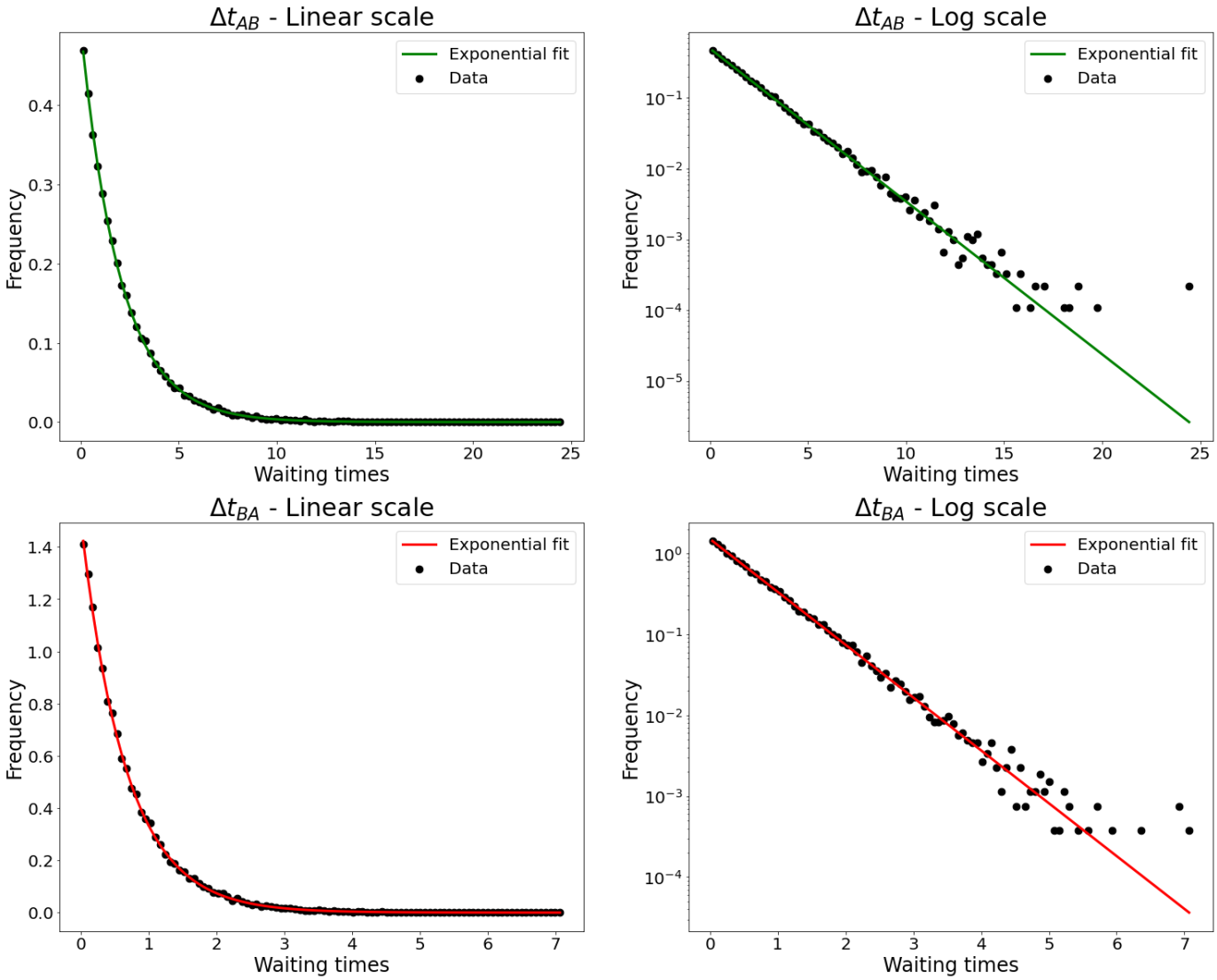
Now, let's look at the statistics of the time between transitions during a single realization of the process. Let's define Δt_{AB} as the time the system spends on state A before transitioning to B, and Δt_{BA} as the time the system spends on state B before transitioning to A. First, let's define a function to calculate this, still using a brute force integration algorithm.

In order to have better statistics, let's consider a longer realization of the experiment, for a time of 10^5 t.u.



One can see these distribution really behave as exponential, as one would expect. Let's fit them and try to recover the transition rates.

```
Exponential fit parameter for state A waiting times: w_01 = [0.4973042]
Exponential fit parameter for state B waiting times: w_10 = [1.50326923]
<matplotlib.legend.Legend at 0x7f37594a4b20>
```

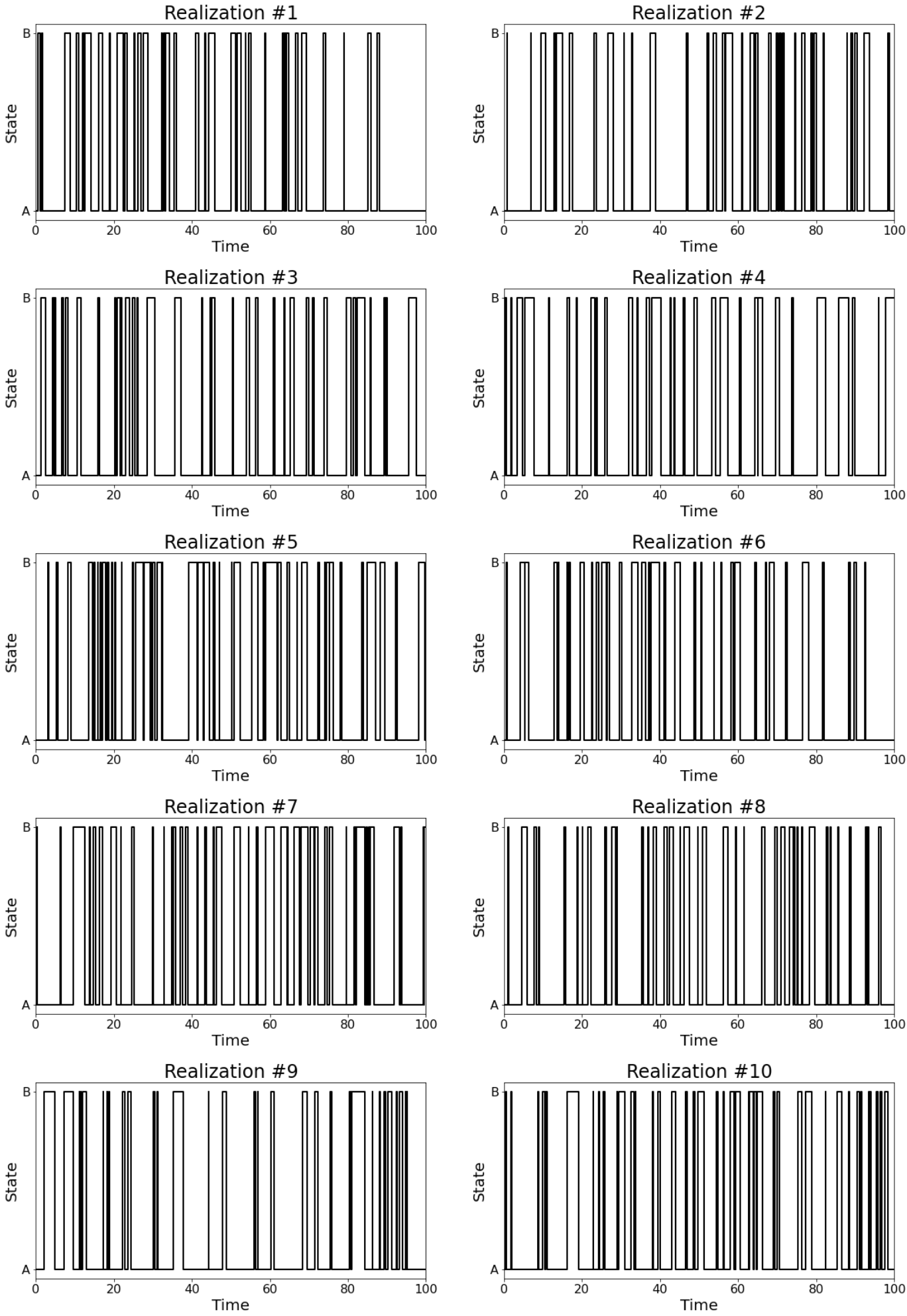


The transition rates obtained from the fit $\omega_{AB} = 0.497$ and $\omega_{BA} = 1.503$ are in good agreement with the actual transition rates we set for the process.

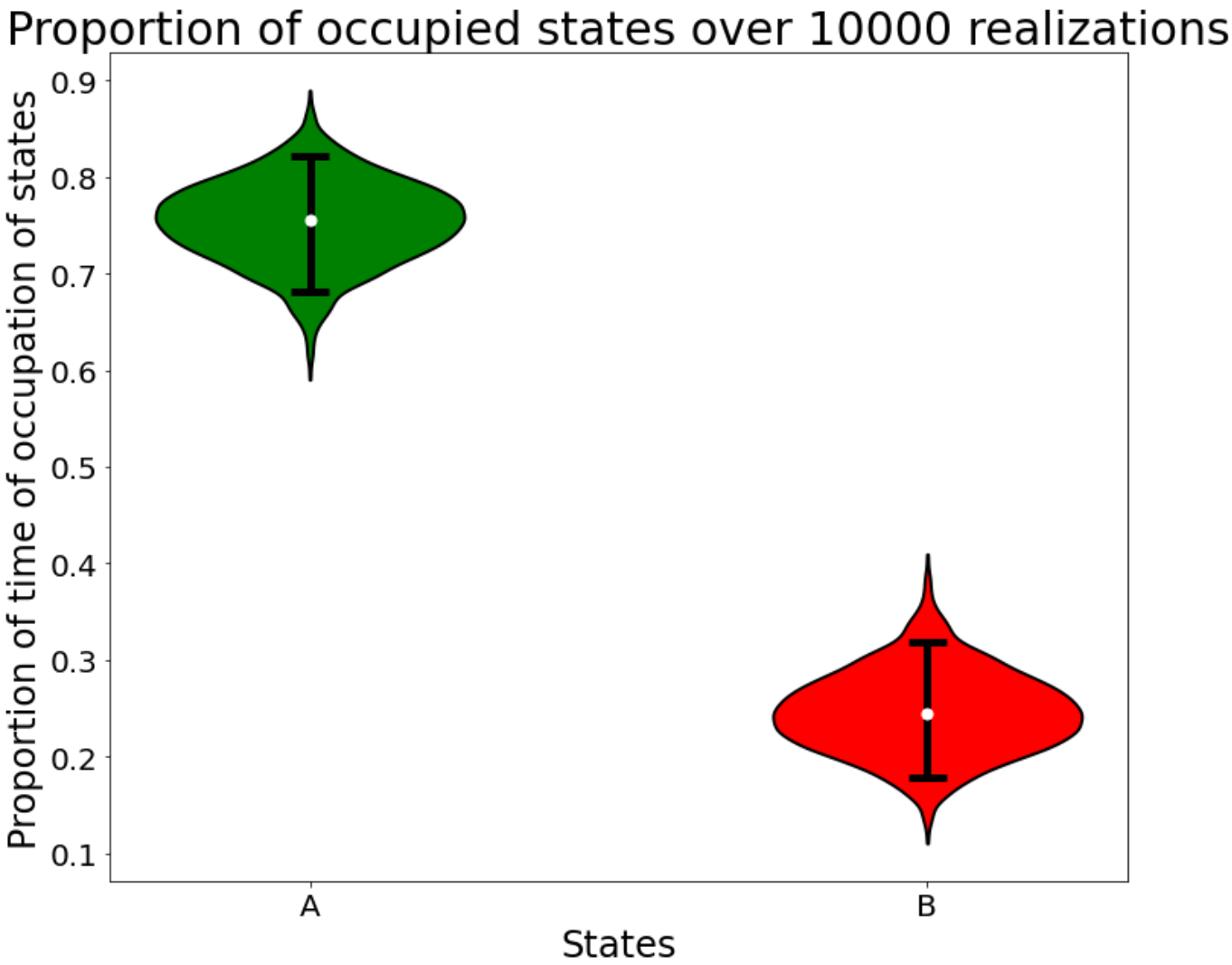
First Reaction Method

Let's now implement the same stochastic process, but using a different, optimized algorithm: the first reaction method. It's implementation is below:

We can now compare the realizations to the ones we've seen generated via brute force:



The realizations look the same, but what about the state occupation statistics?



The state occupation statistics over multiple runs also looks the same.

Now, let's compare the performance of the first reaction method to the one of the brute force algorithm. Let's generate 10 realization with each algorithm and check how long it takes, and let's repeat it five times, to check if there's any fluctuation.

```
Time taken by brute force algorithm for 10 realizations: 0.016721
Time taken by first reaction method for 10 realizations: 0.000157

First reaction method is 70.368 times faster than brute force algorithm.

--------------------------------------------------------------------------------
-------------

Time taken by brute force algorithm for 10 realizations: 0.016721
Time taken by first reaction method for 10 realizations: 0.000115

First reaction method is 113.795 times faster than brute force algorithm.

--------------------------------------------------------------------------------
-------------

Time taken by brute force algorithm for 10 realizations: 0.016721
Time taken by first reaction method for 10 realizations: 0.000113

First reaction method is 168.292 times faster than brute force algorithm.

--------------------------------------------------------------------------------
-------------

Time taken by brute force algorithm for 10 realizations: 0.016721
```

```
Time taken by first reaction method for 10 realizations: 0.000116

First reaction method is 106.154 times faster than brute force algorithm.

--------------------------------------------------------------------------------
------------

Time taken by brute force algorithm for 10 realizations: 0.016721
Time taken by first reaction method for 10 realizations: 0.000113

First reaction method is 124.553 times faster than brute force algorithm.

--------------------------------------------------------------------------------
------------
```

> The first reaction method is clearly faster, but there's some fluctuation to how much. Let's increase the number of realizations to 10^3

```
Time taken by brute force algorithm for 10 realizations: 0.016721
Time taken by first reaction method for 10 realizations: 0.007477

First reaction method is 147.204 times faster than brute force algorithm.

--------------------------------------------------------------------------------
------------

Time taken by brute force algorithm for 10 realizations: 0.016721
Time taken by first reaction method for 10 realizations: 0.007295

First reaction method is 148.521 times faster than brute force algorithm.

--------------------------------------------------------------------------------
------------

Time taken by brute force algorithm for 10 realizations: 0.016721
Time taken by first reaction method for 10 realizations: 0.007411

First reaction method is 146.967 times faster than brute force algorithm.

--------------------------------------------------------------------------------
------------

Time taken by brute force algorithm for 10 realizations: 0.016721
Time taken by first reaction method for 10 realizations: 0.007718

First reaction method is 140.955 times faster than brute force algorithm.

--------------------------------------------------------------------------------
------------

Time taken by brute force algorithm for 10 realizations: 0.016721
Time taken by first reaction method for 10 realizations: 0.007365

First reaction method is 148.462 times faster than brute force algorithm.

--------------------------------------------------------------------------------
------------
```

> The results seem to have stabilized, and we can see that the first reaction method is more than 100 times faster than the brute force algorithm.

Problem 2

Analytic solution for the average

Now we are going to consider another two-state stochastic process, with the same transition rates $\omega_{AB} = 0.5$ and $\omega_{BA} = 1.5$, but now with $N = 1000$ particles instead of only one, with an initial condition of 500 particles in state A, and another 500 in B

First, let's calculate analytically the time evolution of the average number of particles in A. If we have n particles in A, we know that the transition rate to $n-1$ should be $\Omega(n \to n-1) = n\omega_{AB}$, and the rate from n to $n+1$ is $\Omega(n \to n+1) = (N-n)\omega_{BA}$. We can then write the master equation for this process:

$$\frac{\mathrm{d}P(n,t)}{\mathrm{d}t} = (E-1)\left[\omega_{AB} n P(n,t)\right] + \left(E^{-1}-1\right)\left[\omega_{BA}(N-n)P(n,t)\right], \tag{1}$$

where the operator E is such that $E^m f(n) = f(n+m)$.

Let's calculate the average value $\langle n(t)\rangle$, by multiplying the master equation by n, and summing for $n = 0, 1, 2, \ldots, N$:

$$\begin{aligned}\sum_{n=0}^{N} n\frac{\mathrm{d}P(n,t)}{\mathrm{d}t} =& \sum_{n=0}^{N} \omega_{AB} n(n+1)P(n+1,t) + \sum_{n=0}^{N} \omega_{BA} n(N-n+1)P(n-1,t) + \\ &- \sum_{n=0}^{N} \left(\omega_{AB} n^2 + \omega_{BA} n(N-n)\right) P(n,t).\end{aligned}$$

This can be rewritten as:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\sum_{n=0}^{N} nP(n,t) =& \sum_{n=0}^{N} \omega_{AB} n(n-1)P(n,t) + \sum_{n=0}^{N} \omega_{BA}(n+1)(N-n)P(n,t) + \quad (4)\\ &- \sum_{n=0}^{N} \left(\omega_{AB} n^2 + \omega_{BA} n(N-n)\right) P(n,t), \quad (5)\end{aligned}$$

and simplified to

$$\frac{\mathrm{d}}{\mathrm{d}t}\langle n(t)\rangle = \sum_{n=0}^{N} \left[-\omega_{AB} n + \omega_{BA}(N-n)\right] P(n,t). \tag{6}$$

Finally, we get the differential equation for the average number of particles in A:

$$\frac{\mathrm{d}}{\mathrm{d}t}\langle n(t)\rangle = \omega_{BA} N - \omega \langle n(t)\rangle, \tag{7}$$

with $\omega = \omega_{AB} + \omega_{BA}$. To solve this equation, we can define $x(t) = \langle n(t)\rangle - \frac{\omega_{BA}}{\omega} N$, such that:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = -\omega x \Rightarrow x(t) = Ae^{-\omega t} \tag{8}$$

and the solution for the number of particles must then be:

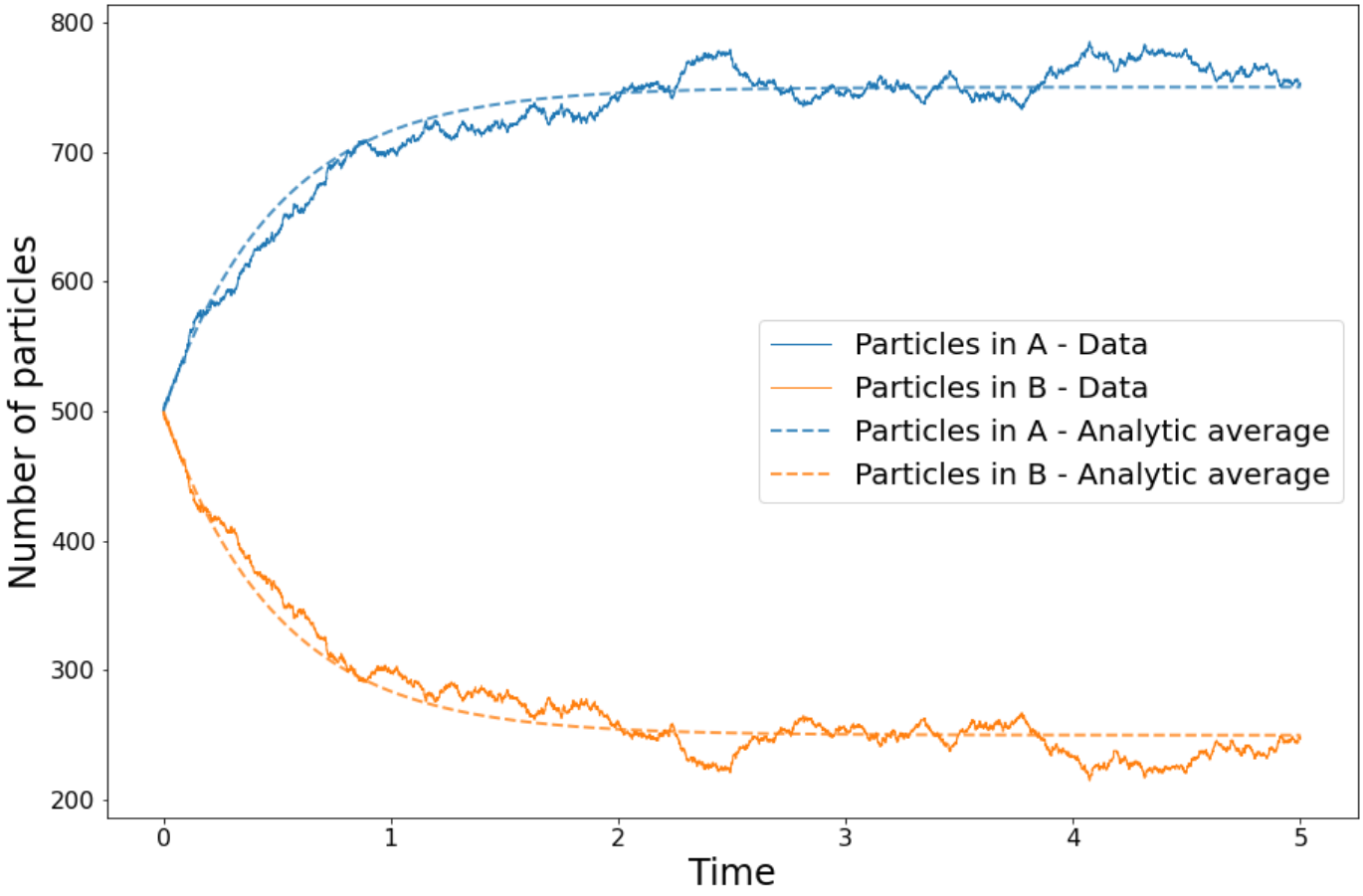
$$\langle n(t)\rangle = \frac{\omega_{BA}}{\omega} N\left(1 - e^{-\omega t}\right) + \langle n(0)\rangle e^{-\omega t}. \tag{9}$$

Simulation

New we are going so simulate this stochastic process, again using the first reaction method, with a little difference form before, as we are now considering multiple particles. Let's write the function for the time evolution of the system:

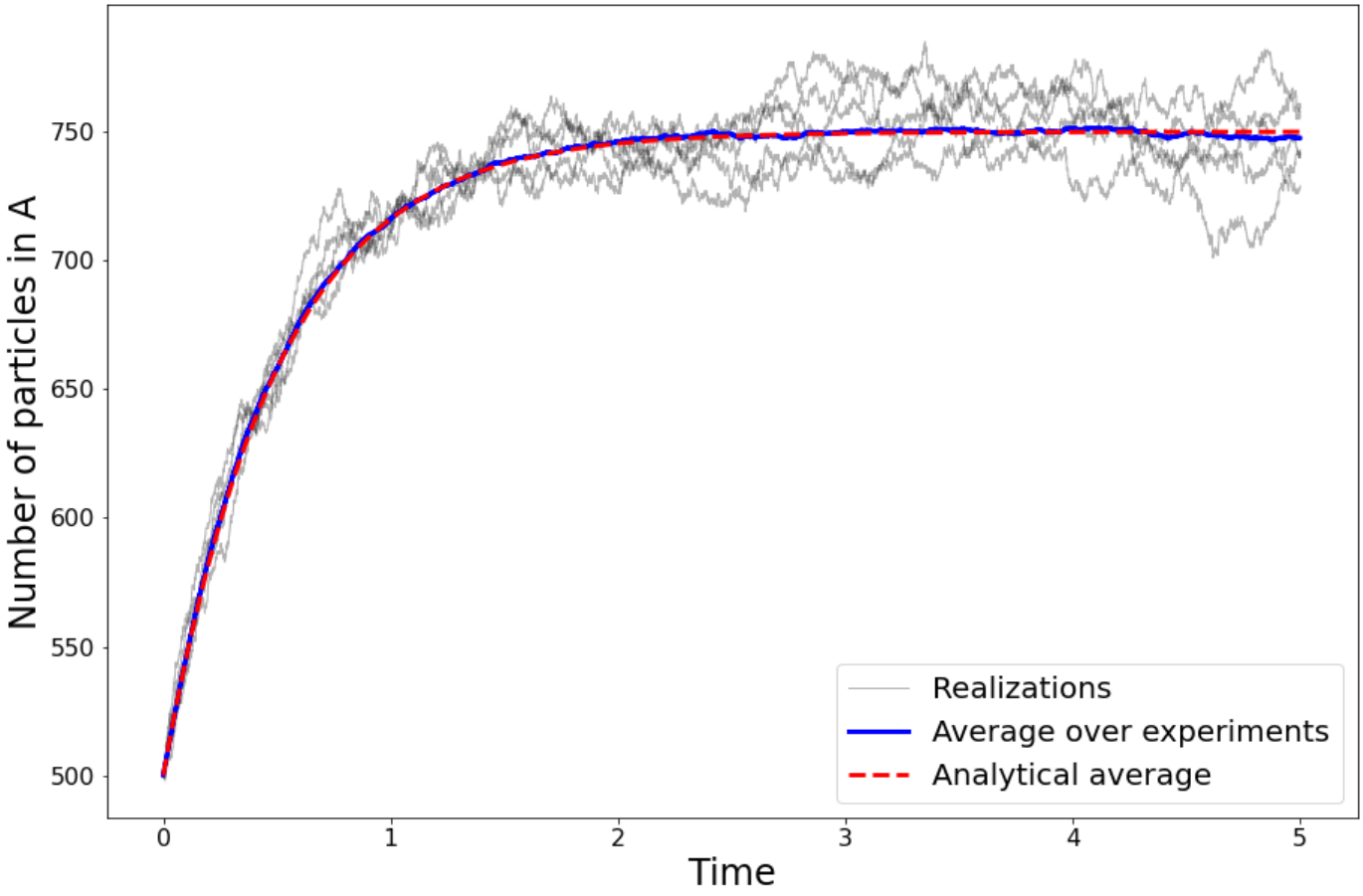
Let's give it a try, and compare the results to the analytically derived expression for the average:

```
Text(0, 0.5, 'Number of particles')
```



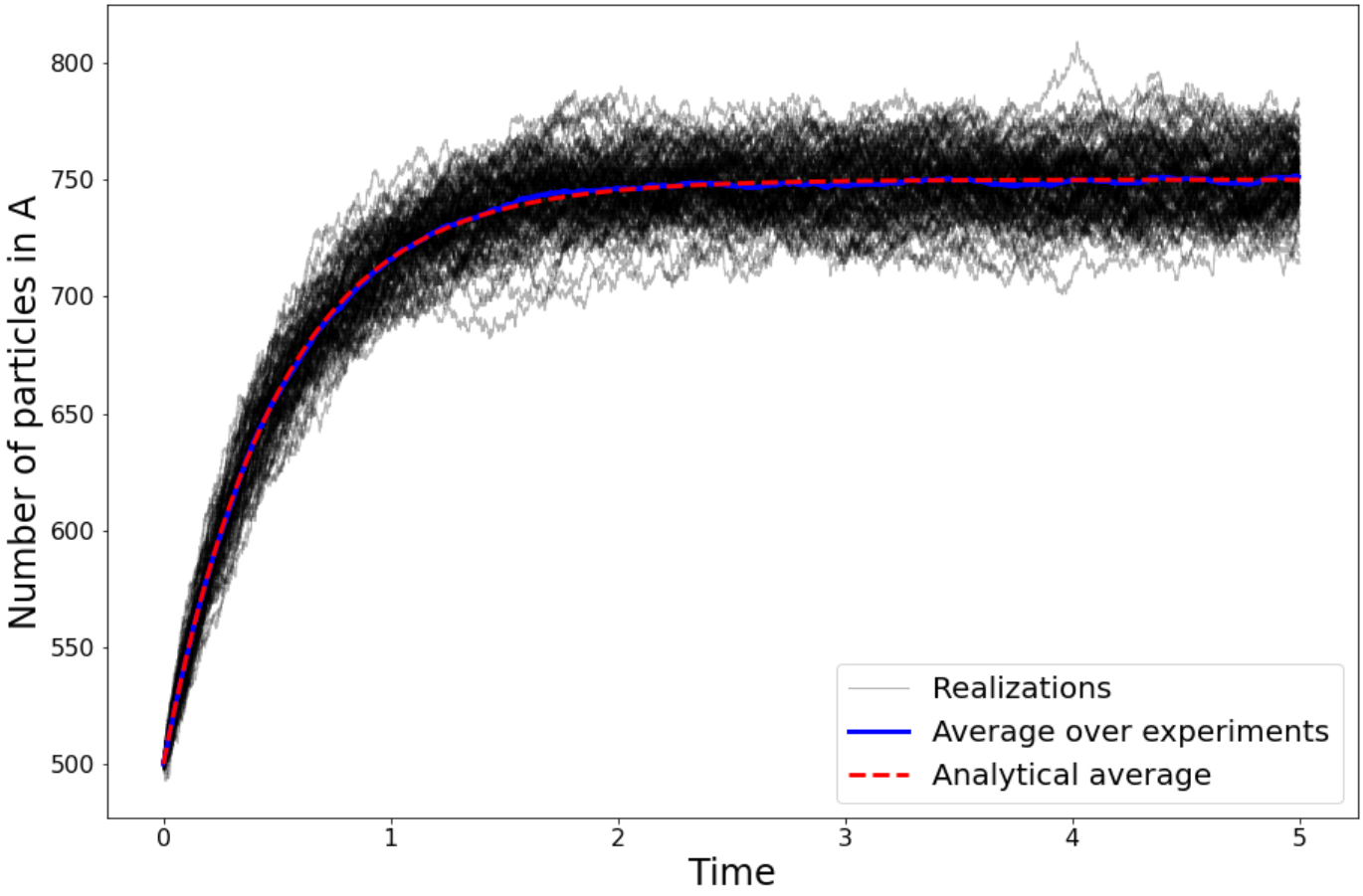
As we can see, the stochastic process fluctuates aroung the expected average. But let's calculate the average from an ensemble of simulations, and see how it relates to our analytical expectation. As the time points of each execution are stochastic, we will interpolate the results to a same time mesh, so that we can calculate averages. We will take the average over 100 realizations of our experiment, and plot the average, together with the analytical solution, and five of the realizations themselves so we can have an idea of the fluctuations.

```
100%|██████████| 100/100 [00:00<00:00, 1106.05it/s]
Text(0, 0.5, 'Number of particles in A')
```



We can see a very good correspondence between the numerical and the analytical averages, as we should expect. We can also look at all of the realizations, forming a "cloud" of curves around the average.

```
100%|██████████| 100/100 [00:00<00:00, 526.57it/s]
Text(0, 0.5, 'Number of particles in A')
```


Number of particles in A
800
750
700
650
600
550
500
0
1
2
3
4
5
Time
Realizations
Average over experiments
Analytical average