NC STATE UNIVERSITY

Monte Carlo Simulation

FIM 548-001

February 15, 2025

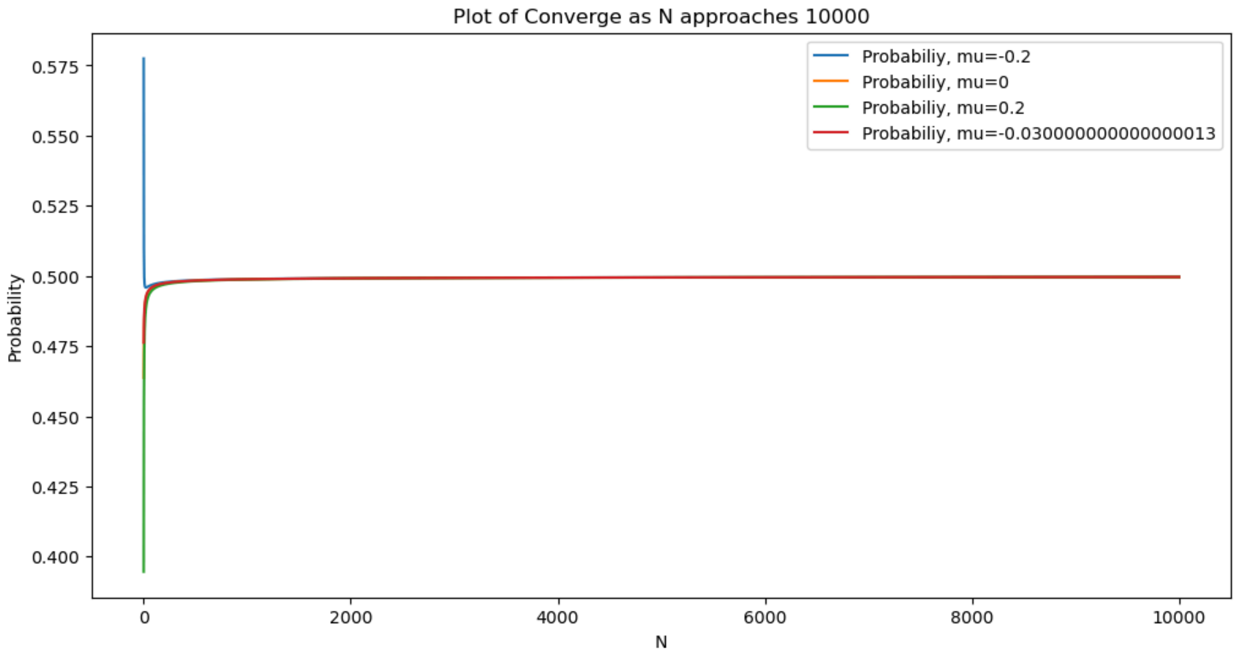
Jinjia Peng

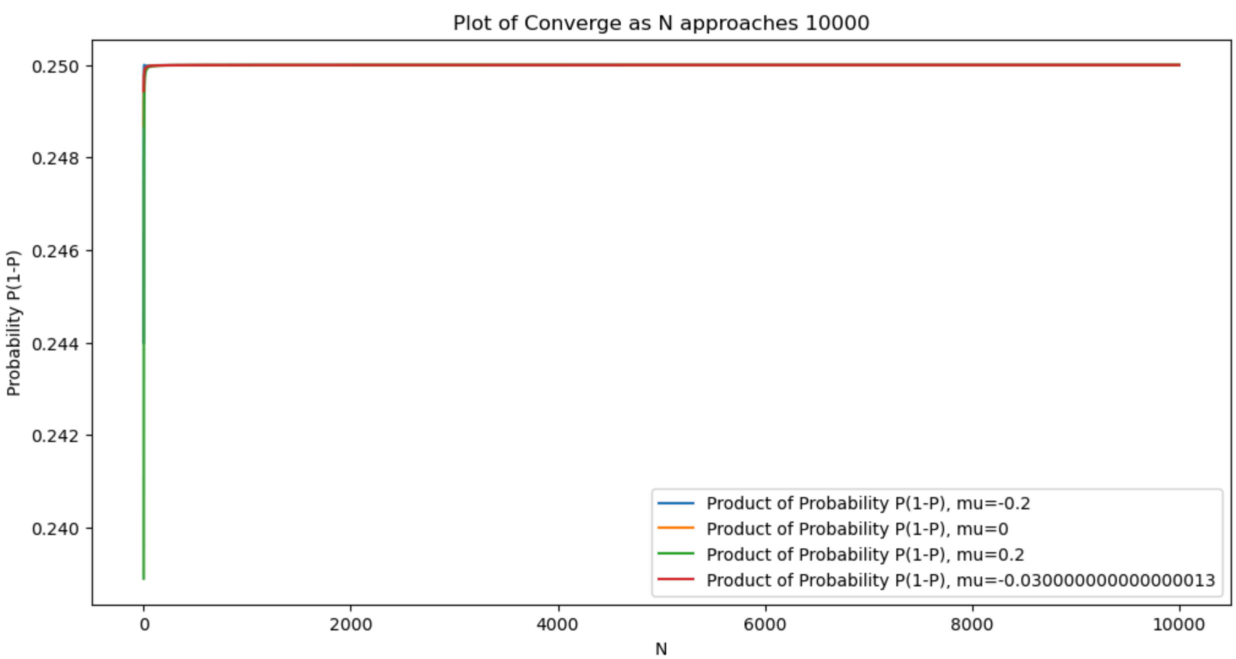
NCSU ID: 200604130

Problem 1

1.

The task is to show the convergence of PN and $PN(1 - PN)$ as $N \to \infty$. Using $N = 10,000$, the result is as the following:



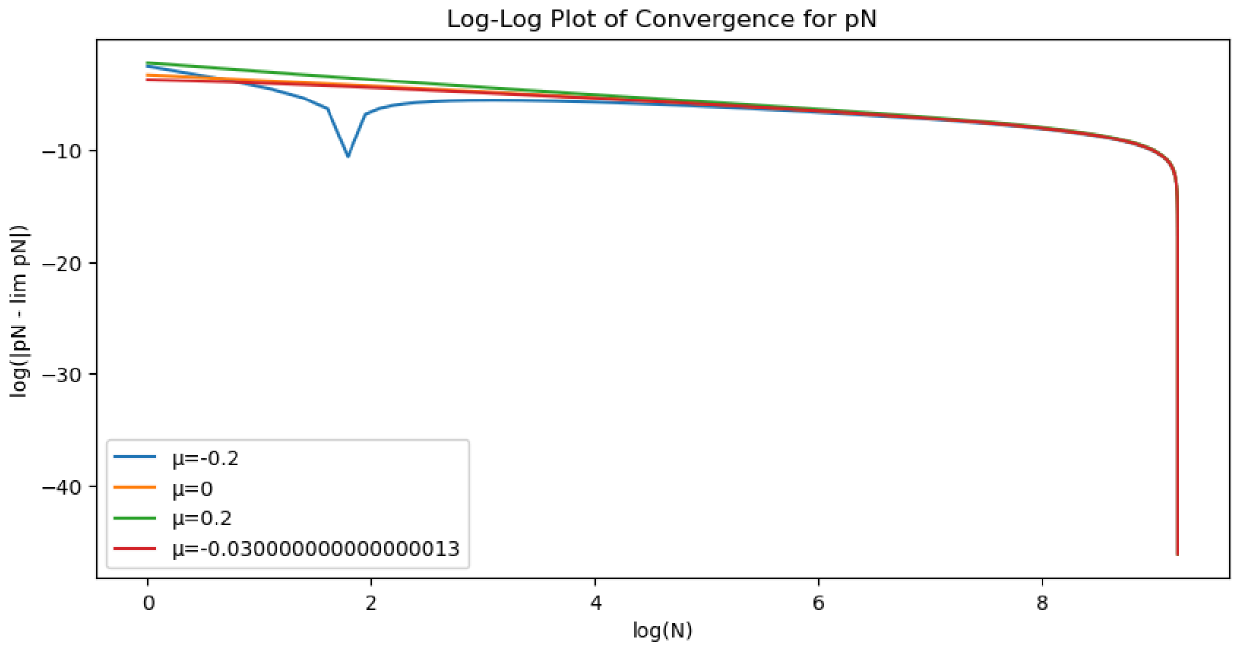


For $\mu = -0.2$:
Estimated limit of pN: 0.49963314192897124
Estimated limit of pN(1 - pN): 0.24999986541515573
For $\mu = 0$:
Estimated limit of pN: 0.49962500122916453
Estimated limit of pN(1 - pN): 0.24999985937592187
For $\mu = 0.2$:
Estimated limit of pN: 0.4996168917804245
Estimated limit of pN(1 - pN): 0.24999985322809212
For $\mu = -0.030000000000000013$:
Estimated limit of pN: 0.4996262203304219
Estimated limit of pN(1 - pN): 0.24999986028875862

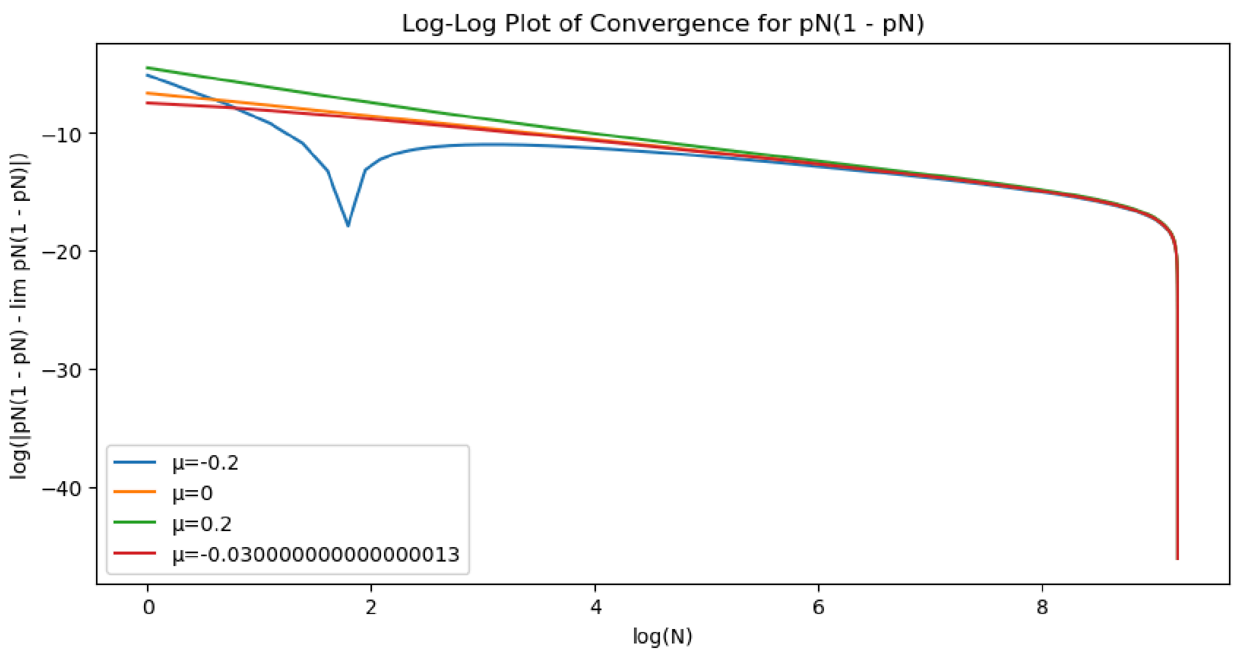
Conclusion: the result of the convergence **does not depend on μ.**

2.

The goal here is to plot the graph (log-log) of the log of the N-step period and the log of the difference between the estimated probability and the final convergent value. The graphical result is as follows:



Log-Log Plot of Convergence for pN
log(|pN - lim pN|)
−10
−20
−30
−40
0
2
4
6
8
log(N)
μ=-0.2
μ=0
μ=0.2
μ=-0.030000000000000013



Log-Log Plot of Convergence for pN(1 - pN)
log(|pN(1 - pN) - lim pN(1 - pN)|)
−10
−20
−30
−40
0
2
4
6
8
log(N)
μ=-0.2
μ=0
μ=0.2
μ=-0.030000000000000013

To estimate the value of k1 and k2,

$$ln\left|pN - \lim_{N\to\infty} pN\right| \approx ln(c_1) - k_1 ln(N)$$

$$ln\left|pN(1 - PN) - \lim_{N\to\infty} pN(1 - pN)\right| \approx ln(c_2) - k_2 ln(N)$$

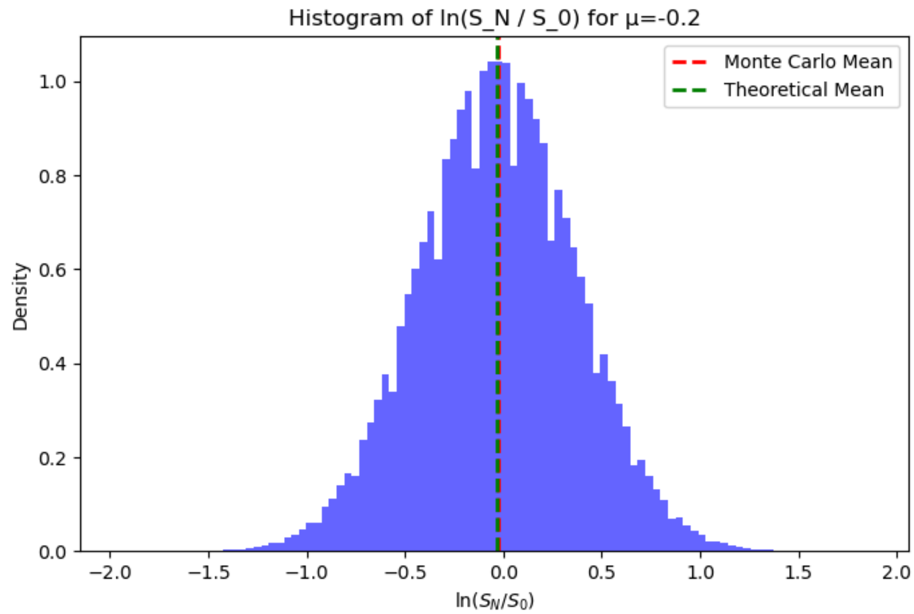
Then, use the curve_fit( ) function in Python, the result is shown below:

For μ = -0.2:
Estimated k1: 0.72, C1: 0.05
Estimated k2: 3.52, C2: 0.01
For μ = 0:
Estimated k1: 0.58, C1: 0.04
Estimated k2: 0.98, C2: 0.00
For μ = 0.2:
Estimated k1: 0.71, C1: 0.10
Estimated k2: 1.49, C2: 0.01
For μ = -0.030000000000000013:
Estimated k1: 0.55, C1: 0.03
Estimated k2: 0.81, C2: 0.00

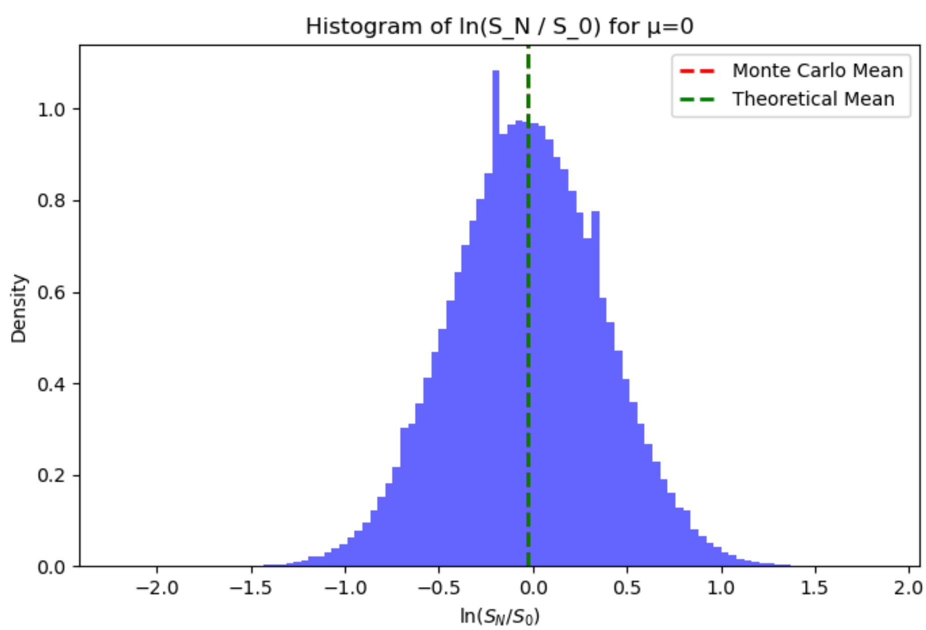
3.

After running the simulation with 1,000,000 samples the distribution is shown in the histograms below:

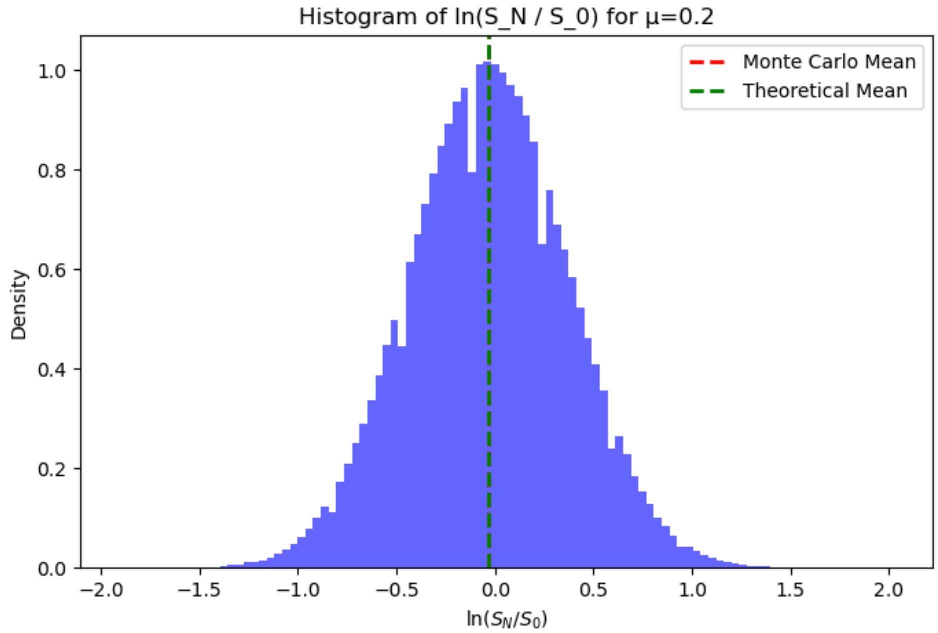
For μ = -0.2:
  Monte Carlo Expected Value: -0.028835, Theoretical: -0.030000
  Monte Carlo Std Dev: 0.397982, Theoretical: 0.400000



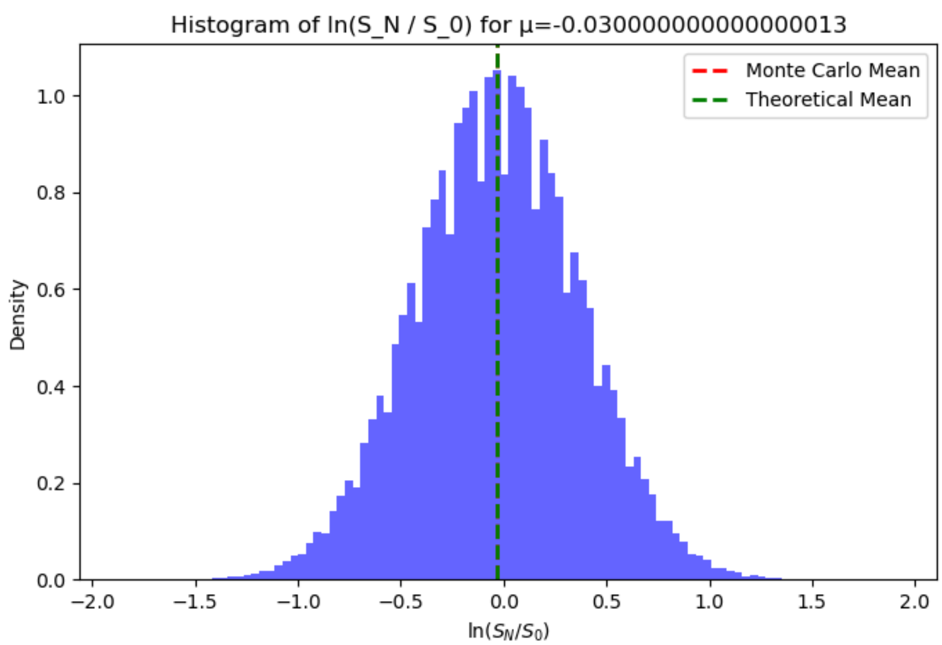
For μ = 0:
  Monte Carlo Expected Value: -0.030062, Theoretical: -0.030000
  Monte Carlo Std Dev: 0.400468, Theoretical: 0.400000



For μ = 0.2:
Monte Carlo Expected Value: -0.030965, Theoretical: -0.030000
Monte Carlo Std Dev: 0.402065, Theoretical: 0.400000



For μ = -0.030000000000000013:
Monte Carlo Expected Value: -0.029925, Theoretical: -0.030000
Monte Carlo Std Dev: 0.399713, Theoretical: 0.400000



Conclusion:

For the distribution of $ln(\frac{SN}{S0})$ after a simulation of large sample size = 1,000,000,

$E[ln(\frac{SN}{S0})] \rightarrow r - \frac{\sigma^2}{2} = -0.03$, $SD[ln(\frac{SN}{S0})] \rightarrow \sigma = 0.4$

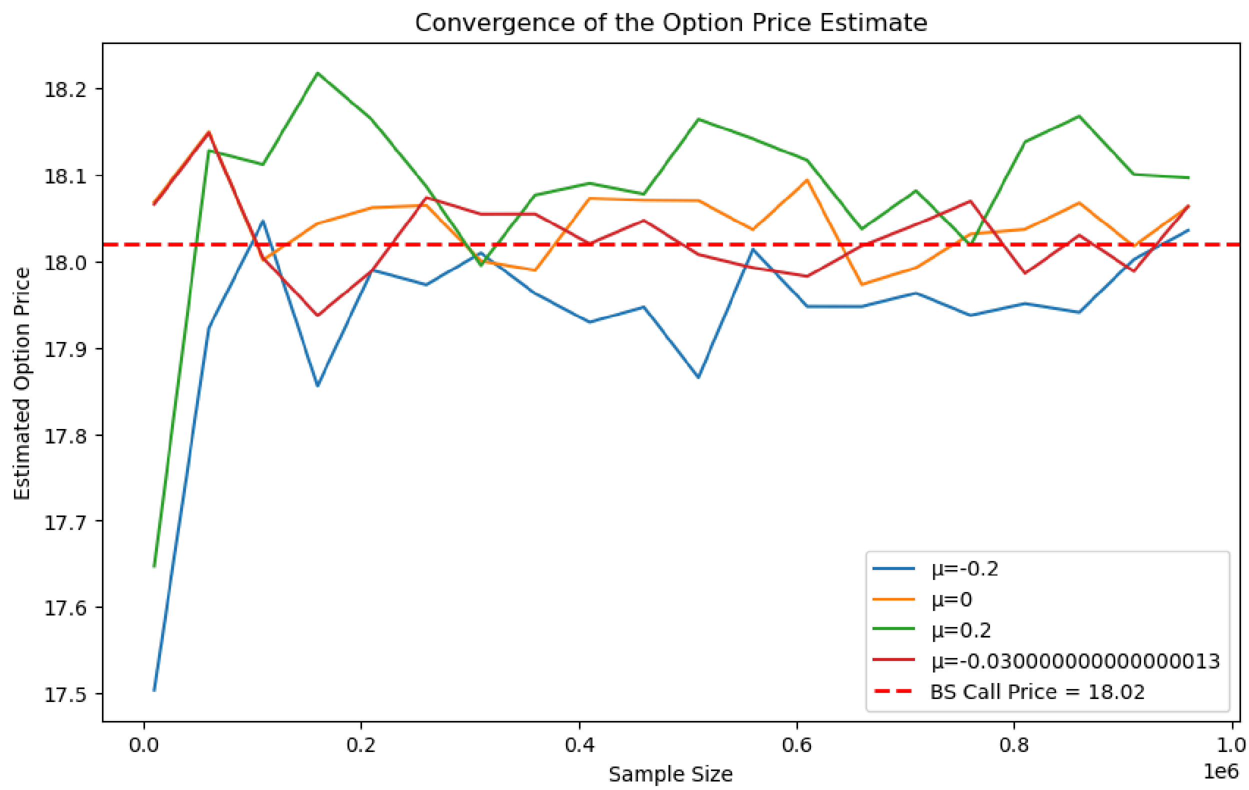
The result is independent from μ.

4.

From part 1 and part 2, observe that P[u] and P[d] converge in such a way that prices process transitions to a log-normal distribution as $N \rightarrow \infty$. As result, $ln(\frac{SN}{S0}) \rightarrow N(r - \frac{\sigma^2}{2}, \sigma^2)$. Therefore, the conclusion is that the binomial model converges to a Geometric Brownian Motion as $N \rightarrow \infty$. In the previous parts, there is no significance dependence of $ln(\frac{SN}{S0})$ on μ. As a result, under a risk-neutral approach, asset prices follow a drift directed by r, not μ.

5.

Plotting the simulation for different μ values, the graph :



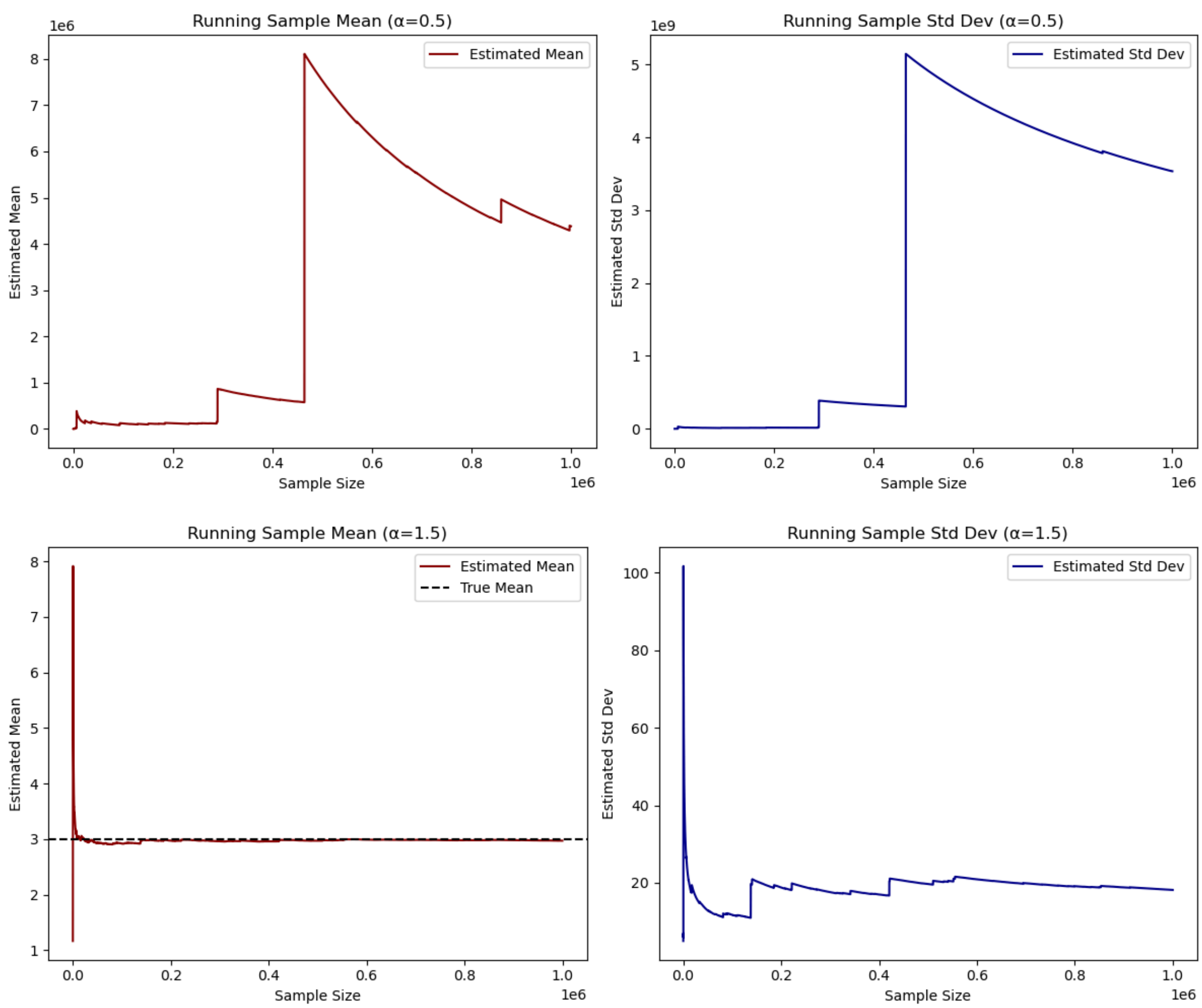
indicates that the European Call option prices converge to the Black-Scholes' solution (18.02). This further verifies the statement that when N is large, the convergence of the expected value of the discounted option payoff $E[e^{-r}(s_1^{\ N} - K)^+]$ is independent of μ.

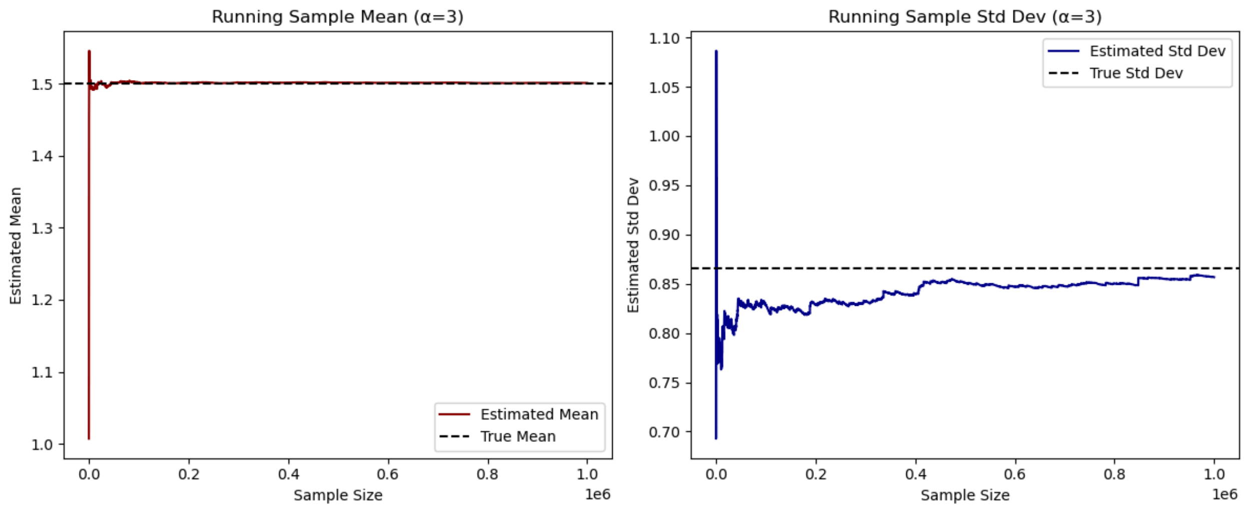
Problem 2

Implementing the algorithm (see code), in the experiment, the proportion of times the true mean falls within the confidence interval: 0.9504

Problem 3

1.

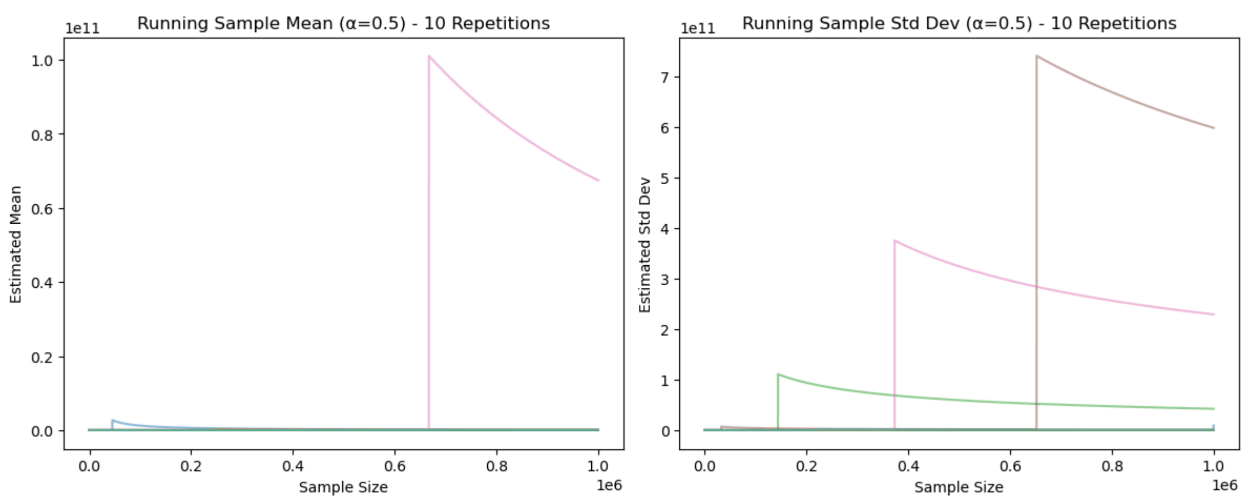


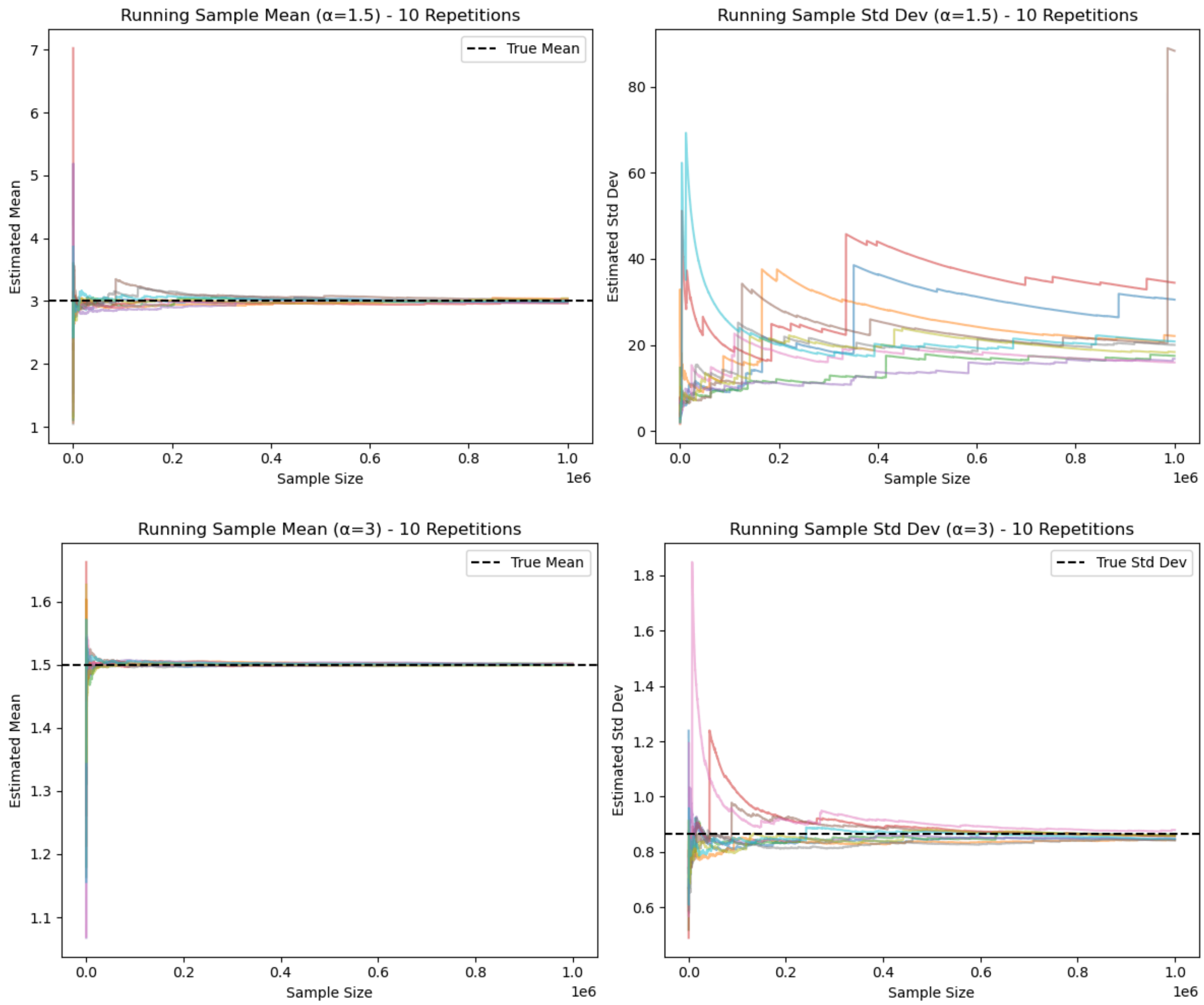


Notice that the the expected value of the random variables that follows the Pareto distribution diverges if the parameter $\alpha \le 1$and the variance diverges if $\alpha \le 2$.

This property is presented in the graphs that when α = 0.5, both the sample mean and the sample standard deviation diverges. When α = 1.5, the sample mean converges to the true mean as N becomes very large, but the sample standard deviation diverges. Finally, when α = 3, both the sample mean and the sample standard deviation converge as N becomes very large.

2.

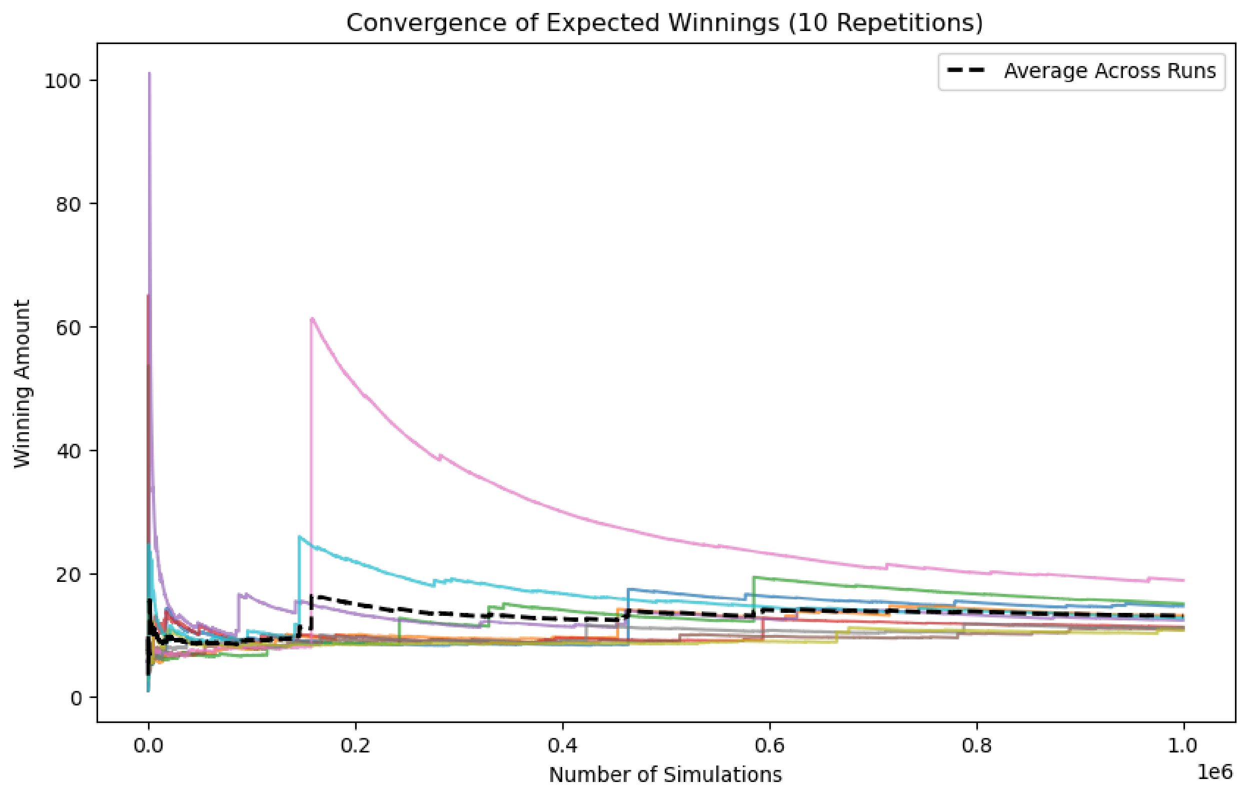




This repeated simulation (10 times), further confirms the statement made in part 1. Indeed, the convergence and divergence of the sample mean and standard deviation depend on the value of α

Problem 4

Suppose the coin has probability p of flipping a tail, then the flipping result follows a geometric distribution that the probability of the tail occurs at the n-th flip is $P[X = n] = (1 - p)^{n-1}p$
The amount of cash winning per game is 2^{n-1}, where n is defined previously.
Now, plotting the moving expected winning amount as the number simulation increases. 10 Different simulations have been conducted here. Notice that the mean of the moving expected winning amount of all 10 simulations (black dashed line) lies under \$20.



Run 1: Estimated Expected Winnings: \$10.46
Run 2: Estimated Expected Winnings: \$13.07
Run 3: Estimated Expected Winnings: \$17.02
Run 4: Estimated Expected Winnings: \$18.25
Run 5: Estimated Expected Winnings: \$17.65
Run 6: Estimated Expected Winnings: \$14.98
Run 7: Estimated Expected Winnings: \$9.41
Run 8: Estimated Expected Winnings: \$9.72
Run 9: Estimated Expected Winnings: \$10.75
Run 10: Estimated Expected Winnings: \$11.12

Overall Estimated Expected Winnings (Averaged Across 10 Runs): \$13.25

Assuming a fair coin, the probability that a tail result happening at the nth time of the flip is

$$P[X = n] = (1 - p)^{n-1}p = 0.5^{n}$$

And the expected value of winning is $E[winning\ amount] = 2^{n-1} * 0.5^{n} = 0.5$, note that it is a positive expected value, meaning that if a very large number of games were played, the winning amount approaches infinity. Therefore, it outperforms the simulation result. However, The graph indicates that it is safe to take the $20 because the majority of the simulations lie under it, and the extreme winning cases will quickly be adjusted back to a low level as N increases..

Problem 5

1.

Given that:

$g(x) = \lambda e^{-\lambda x}$

The A-R method requires:

$f(x) = \frac{1}{6}x^{3}e^{-x} \leq c(\lambda)\lambda e^{-\lambda x}\ for\ all\ x \geq 0.$

$\Rightarrow c(\lambda) \geq \frac{f(x)}{g(x)} = \frac{1}{6\lambda}x^{3}e^{(\lambda-1)x}$

The goal is to find the maximum of $\frac{f(x)}{g(x)}$ to determine the minimum value of $c(\lambda)$.

Taking the first derivative of $\frac{1}{6\lambda}x^{3}e^{(\lambda-1)x}$, yields $\frac{1}{6\lambda}e^{(\lambda-1)x}(3x^{2} + x^{3}(\lambda - 1))$. Set this equation equal to 0, solve for x, then x has value $\frac{3}{-(\lambda-1)}$, which will be the maximum point of $\frac{f(x)}{g(x)}$. Plugging in this value, generates $c(\lambda) = \frac{9e^{-3}}{2(1-\lambda)^{3}\lambda}$.

2.

The goal is to minimize $c(\lambda)$. Taking the first derivative, yields $c^{(1)}(\lambda) = \frac{-9e^{-3}(-4\lambda+1)}{2(-\lambda-1)^{4}\lambda^{2}}$. Setting the first derivative to be zero, $\Rightarrow -4\lambda + 1 = 0 \quad \Rightarrow \lambda = 0.25$ will be the optimal most efficient parameter with c = 1.062 for the A-R method.

Random Variable Generating Process:

1. Choose c = 1.062 as the parameter
2.Generate $U \sim Uniform(0, 1)$independently of X.
3.Generate a random variable $X \sim Exp(0.25)$
4. Accept X as a sample of the original distribution if $cU \leq \frac{f(X)}{g(X)}$, else, reject X and repeat the process from the beginning to get a new X until the number of generations is satisfied.

Problem 6

After running the simulations, the estimation the expectation and variance of the total number of hits is shown below:

	N (Simulations)	Estimated Mean Hits	Estimated Variance
0	10	1.300000	1.788889
1	100	0.940000	0.925657
2	1000	1.046000	1.040925
3	10000	1.000700	1.006800
4	100000	1.000920	1.004789
5	1000000	0.999135	0.999225