MFE 405 Project 2
Yong Jia (Justin) Tan

1. Generate Bivariate-Normally Distributed Random Vectors

When generating multi-dimensional normal vectors of the following distribution:
$X \sim N(\mu, \Sigma)$,
where $\mu = (\mu_1 \dots \mu_n)'$, $\Sigma = (\sigma_{ij})_{i,j=1}^{n}$,
We can use the Cholesky decomposition of the variance-covariance matrix:
$\Sigma = L \cdot L'$,
and define $X = \mu + L \cdot Z$,
where $Z = (z_1, \cdots, z_n)'$ is a column vector of i.i.d. standard normal values.
In this case where n = 2, we can generate draws from bivariate-normally distributed random distributions with the following:
$X = \mu_X + \sigma_X z_1$
$Y = \mu_Y + \sigma_Y \rho z_1 + \sqrt{1-\rho^2}\sigma_Y z_2$
Using these results and simulating 1,000 values from this distribution, we compute $\rho(a) = -0.7005$, which is very close to the theoretical value of -0.7.

2. Expected Value – Bivariate-Normal

To calculate the expected value of the equation, we generate draws from bivariate-normally distributions with the same method from part 1. After simulating, we computed the expected value to be 1.57336.

3. Brownian Motion Simulation, Expected Values, Variance Reduction

To calculate the expected values, we need to first compute random draws of Brownian motion at time t. We can use this following fact:
$W_T = (W_T - W_0) \stackrel{d}{=} z\sqrt{T}$,
where z is a random standard normal number.
Using this we can simulate 20,000 values of $W_5, W_{0.5}, W_{3.2}, W_{6.5}$. Then, using these values, we can compute expected values of the given equations.
Now, we can attempt to use variance reduction techniques to improve our simulation. In this case we try to use control variates. For $E(X) = \mu$, we find a known $E(Y) = \delta$ and define $T = X - \gamma(Y - \delta)$. In this case, $E(T) = \mu$. The key is to minimize $Var(T) = (1-\rho^2)Var(X)$, thus $\gamma^* = \sigma_{XY}/\sigma_Y^2$. For control variates, I chose $sin(U), U \sim [0, \pi]$ for $E(W_5^2 + sin(W_5))$, and $cos(U), U \sim [0, \pi]$ for $E(e^{\frac{t}{2}} cos(W_t))$.
We can algebraically solve δ for both cases:
$E(sin(U)) = 2/\pi$
$E(cos(U)) = 0$

Using the above results, we can attempt to reduce variance. The computed expected values and variances are shown in the table below:

Expected Values				Variances			
Ea1	5.02413	**Eb1**	5.02231	**Va1**	50.1282	**Vb1**	49.8539
Ea2	0.99879	**Eb2**	0.999574	**Va2**	0.127292	**Vb2**	0.114966
Ea3	0.979301	**Eb3**	0.986833	**Va3**	11.3849	**Vb3**	10.2471
Ea4	0.93487	**Eb4**	0.960452	**Va4**	335.346	**Vb4**	322.221

As we can see, after variance reduction by control variates, the variance was consistently decreased, and the precision of the expected value computation improved. If we wish to further reduce variance, we should look for a distribution Y that has an even higher correlation with X.

When looking closer at the values of the last three integrals (for the cases t = 0.5, 3.2, 6.5), we can see that all three values approach 1. If we apply Ito's Lemma on the integral, we can prove that it is actually a martingale, and by taking expectations, we find that the expectation is 1. So, the expected value for all t values is always approaching 1 in our calculations. However, as t increases, the variance increases, and the computed expected value is less accurate. As we know that the variance depends on the length of time t, this result makes sense.

4. European Call Option Price Simulation (Geometric Brownian Motion)

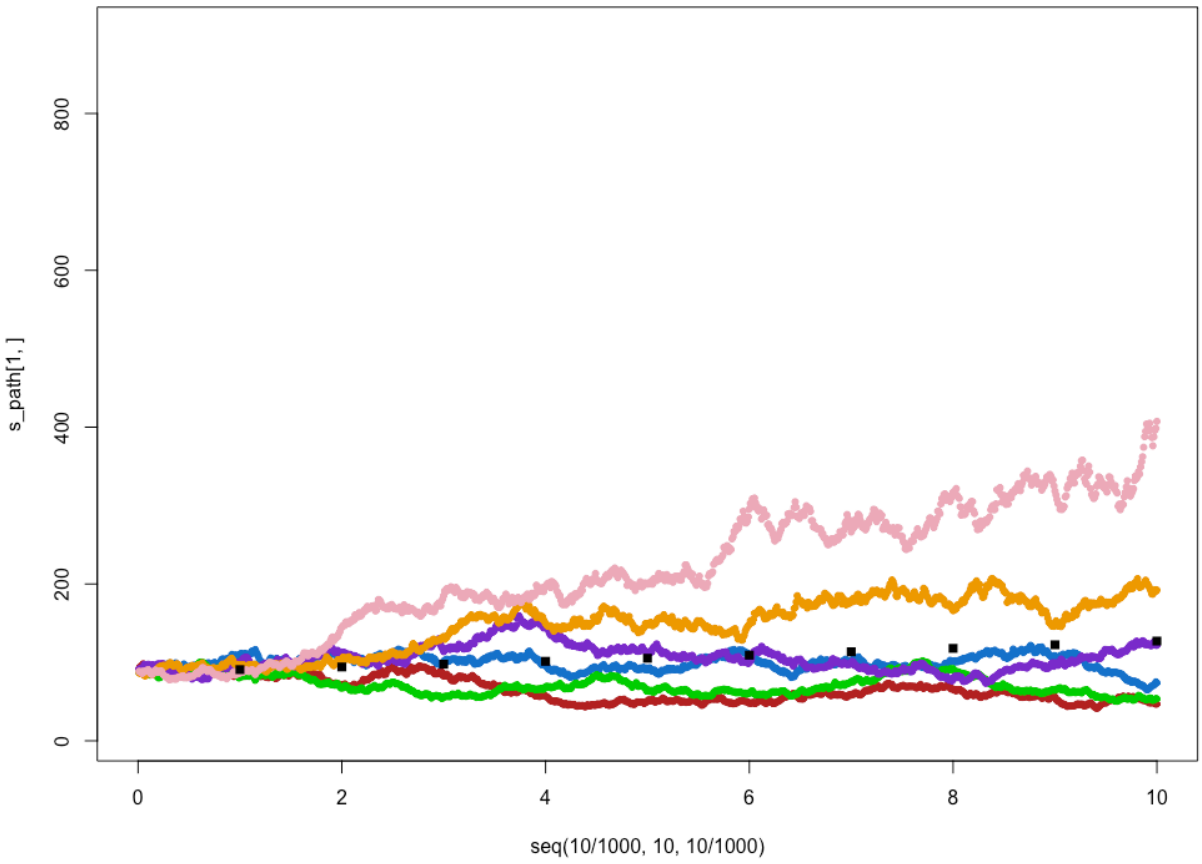
After using similar methods for simulating Brownian motion from part 3, we can estimate the price of a European call option by using Monte Carlo simulation (n = 10,000). We can also again apply control variates to reduce variance. In this case, we set $E(Y) = E(W_5) = 0$ as our control variates. The computation and variance reduction results are summarized in the table below:

European Call Option	Price	Variance
Monte Carlo	18.6089	1055.08
Monte Carlo w/ Variance Reduction	18.5075	388.044

We can see that the variance reduction technique was very successful and did improve precision in price estimation.
Now, we can compare these results with the exact value of the option as given from the Black-Scholes formula:
$C(S_t, t) = N(d_1)S_t - N(d_2)Ke^{-r(T-t)}$, where
$d_1 = \frac{1}{\sigma\sqrt{T-t}}[ln(\frac{S_t}{K}) + (r + \frac{\sigma^2}{2})(T-t)]$
$d_2 = d_1 - \sigma\sqrt{T-t}$
Substituting in the values and solving the equations, we find the final result: $C = 18.28$. Our simulated result is quite close, and if we increase the number of simulations, we might get an even more accurate result.

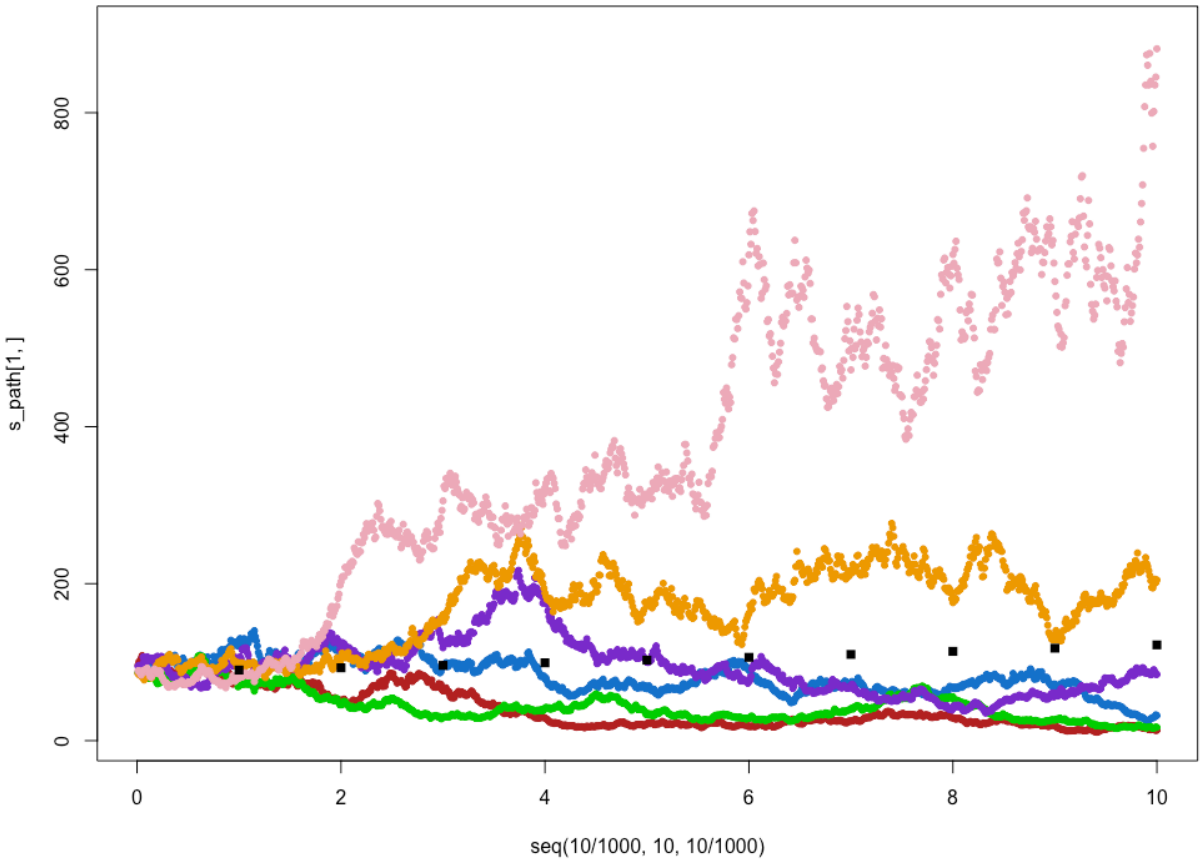
5. Simulate Geometric Brownian Motion Path

First, we generate 1,000 simulation of S_n for n ranging 1-10 using similar methods from part 3 and 4. Then, we instead simulate 6 paths of S_t for $0 \leq t \leq 10$, diving the interval [0,10] into 1,000 parts. The simulated paths, as well as $E(S_n)$, are plotted as below:



In this plot, the black squares show the empirical $E(S_n)$. We see that the paths for the geometric Brownian motion is random and can go either up or down. We also see that our simulation method provides a good estimate of such paths.

If we were to increase σ from 18% to 35%, the graph for $E(S_n)$ and the simulated paths would instead look like the one below:



We can obviously see that although the paths in both cases are random and show movements up and down, the second graph's magnitude of movements is much larger due to much larger volatility. We also observe that $E(S_n)$ does not change.

6. Computing Integral: Euler's Method, Monte Carlo, Importance Sampling

Using Euler's Method to estimate the integral, we split the integral into 1,000 small "rectangles" and we find the result to be 3.13956, which is not too far off from the value of π.

Using Monte Carlo simulation, we first generate 1,000 random uniformly distributed numbers on [0,1], then calculate the expected value of the expression inside the integral. This yields a value of 3.16827, which yet again is not far off from π.

In order to reduce variance and improve our results, we use importance sampling. Importance sampling changes the integral as following:

$$\mu = E_f g(x) = \int g(x)f(x)dx = \int \frac{g(x)f(x)}{h(x)}h(x)dx = E_k(\frac{g(x)f(x)}{h(x)})$$

In this case,

$$\mu = 4\int_0^1 \sqrt{1-x^2}dx$$

We can set $h(x) = \frac{1-\alpha x^2}{1-\alpha/3}$, thus we can get:

$$\mu = E_h(\frac{\sqrt{1-U^2}}{\frac{1-\alpha U^2}{1-\alpha/3}})$$

In order to minimize the variance, $\alpha = 0.74$. Using these results, we simulate and estimate the integral (4μ) to be 3.14143, which is even more accurate and has even smaller errors than our previous results.