CFRM 505 Homework 5

Eunki Chung
eunkich@uw.edu

March 9, 2023

Problem 1

Suppose $X \sim U(-1, 1)$ and $Y \sim U(-1, 1)$ are i.i.d. random variables. Consider the probability

$$\mathbb{P}\left[X^2 + 2Y^2 < 2\right]$$

For each part, implement your method using at least $10,000$ samples.

The estimate of the probability is

$$\theta = \mathbb{P}\left[X^2 + 2Y^2 < 2\right] = E[Z]$$

where $Z = \mathbb{1}_{\{X^2+2Y^2<2\}}(X, Y)$

Part 1

Estimate this probability directly by simulating both X and Y. Report your estimate along with a 95% confidence interval.

```python
import numpy as np

N = 10000

X = np.random.uniform(-1, 1, N)
Y = np.random.uniform(-1, 1, N)
Z = (X**2 + 2 * Y ** 2) < 2

zdelta = 1.96
var = np.var(Z)
error = zdelta * np.sqrt(var / N)

print("Direct method: ")
print(f"Estimate: {np.mean(Z):.5f}")
print(f"Confidence Interval: {np.round(np.mean(Z) + np.array([-1, 1]) * error, 5)}")
print("Error bar size = {:.5f}".format(error))
```

```
Direct method:
Estimate: 0.90920
Confidence Interval: [0.90357 0.91483]
Error bar size = 0.00563
```

Part 2

Estimate this probability using conditional Monte Carlo by conditioning on X. Report your estimate along with a 95% confidence interval. How much did this reduce the error from part 1?

Consider using conditional Monte Carlo by conditioning on X.

$$\begin{aligned}
E[Z|X=x] &= P[X^2+2Y^2<2|X=x] \\
&= P\left[Y^2 < \frac{2-x^2}{2}\right] \\
&= P\left[-\sqrt{\frac{2-x^2}{2}} < Y < \sqrt{\frac{2-x^2}{2}}\right] \\
&= F_Y\left(\sqrt{\frac{2-x^2}{2}}\right) - F_Y\left(-\sqrt{\frac{2-x^2}{2}}\right)
\end{aligned}$$

Note that cdf of Y is given by,

$$F_Y(y) = \begin{cases} 0 & \text{if } y < -1 \\ \frac{y+1}{2} & \text{if } -1 \le y < 1 \\ 1 & \text{if } 1 \le y \end{cases}$$

From the support of X, we have,

$$\begin{aligned}
-1 \le x &\le 1 \\
0 \le x^2 &\le 1 \\
0 \ge -x^2 &\ge -1 \\
2 \ge 2-x^2 &\ge 1 \\
1 \ge \frac{2-x^2}{2} &\ge \frac{1}{2} \\
1 \ge \sqrt{\frac{2-x^2}{2}} &\ge \sqrt{\frac{1}{2}} \\
-1 \le -\sqrt{\frac{2-x^2}{2}} &\le -\sqrt{\frac{1}{2}}
\end{aligned}$$

Thus,

$$\begin{aligned}
E[Z|X=x] &= F_Y\left(\sqrt{\frac{2-x^2}{2}}\right) - F_Y\left(-\sqrt{\frac{2-x^2}{2}}\right) \\
&= \frac{\sqrt{\frac{2-x^2}{2}}+1}{2} - \frac{-\sqrt{\frac{2-x^2}{2}}+1}{2} \\
&= \frac{2\sqrt{\frac{2-x^2}{2}}}{2} \\
&= \sqrt{\frac{2-x^2}{2}} \qquad \text{for all } -1 \le x \le 1
\end{aligned}$$

```
X = np.random.uniform(-1, 1, N)
Z = np.sqrt((2-X**2) / 2)

zdelta = 1.96
var = np.var(Z)
error_X = zdelta * np.sqrt(var / N)

print("Conditional Monte Carlo: ")
print(f"Estimate: {np.mean(Z):.5f}")
print(f"Confidence Interval: {np.round(np.mean(Z) + np.array([-1, 1]) * error_X, 5)}")
print("Error bar size = {:.5f}".format(error_X))
print("Error reduction from conditioning on X = {:.5f}%".format((1 - error_X / error) * 100))
```

```
Conditional Monte Carlo:
Estimate: 0.91012
Confidence Interval: [0.90848 0.91176]
Error bar size = 0.00164
Error reduction from conditioning on X = 70.90509%
```

Part 3

Estimate this probability using conditional Monte Carlo by conditioning on Y. Report your estimate along with a 95% confidence interval. How much did this reduce the error from part 1?

Consider using conditional Monte Carlo by conditioning on Y.

$$
\begin{aligned}
E[Z|Y=y] &= P[X^2+2Y^2<2|Y=y] \\
&= P\left[X^2<2-2y^2\right] \\
&= P\left[-\sqrt{2-2y^2}<X<\sqrt{2-2y^2}\right] \\
&= F_X\left(\sqrt{2-2y^2}\right)-F_X\left(-\sqrt{2-2y^2}\right)
\end{aligned}
$$

Note that cdf of X is given by,

$$
F_X(x) = \begin{cases} 0 & \text{if } x<-1 \\ \frac{x+1}{2} & \text{if } -1\le x<1 \\ 1 & \text{if } 1\le x \end{cases}
$$

From the support of Y, we have,

$$
\begin{aligned}
-1 \le y &\le 1 \\
0 \le y^2 &\le 1 \\
0 \ge -2y^2 &\ge -2 \\
2 \ge 2-2y^2 &\ge 0 \\
\sqrt{2} \ge \sqrt{2-2y^2} &\ge 0 \\
-\sqrt{2} \le -\sqrt{2-2y^2} &\le 0
\end{aligned}
$$

Notice that $\sqrt{2-2y^2}>1$ and $-\sqrt{2-2y^2}<-1$ if $|y|<\frac{1}{\sqrt{2}}$. That is,

$$
\begin{aligned}
E[Z|Y=y] &= F_X\left(\sqrt{2-2y^2}\right)-F_X\left(-\sqrt{2-2y^2}\right) \\
&= \begin{cases} \frac{\sqrt{2-2y^2}+1}{2}-\frac{-\sqrt{2-2y^2}+1}{2} & \text{if } |y|\ge\frac{1}{\sqrt{2}} \\ 1-0 & \text{if } |y|<\frac{1}{\sqrt{2}} \end{cases} \\
&= \begin{cases} \sqrt{2-2y^2} & \text{if } |y|\ge\frac{1}{\sqrt{2}} \\ 1 & \text{if } |y|<\frac{1}{\sqrt{2}} \end{cases} \\
&= \min\{\sqrt{2-2y^2},1\}
\end{aligned}
$$

```python
Z = np.minimum(np.sqrt(2 - 2 * Y ** 2), 1)

zdelta = 1.96
var = np.var(Z)
error_Y = zdelta * np.sqrt(var / N)

print("Conditional Monte Carlo: ")
print(f"Estimate: {np.mean(Z):.5f}")
print(f"Confidence Interval: {np.round(np.mean(Z) + np.array([-1, 1]) * error_Y, 5)}")
print("Error bar size = {:.5f}".format(error_Y))
```

```
print("Error reduction from conditioning on Y = {:.5f}%".format((1 - error_Y / error) * 100))
```

```
Conditional Monte Carlo:
Estimate: 0.91013
Confidence Interval: [0.90646 0.9138 ]
Error bar size = 0.00367
Error reduction from conditioning on Y = 34.78709%
```

Problem 2

Suppose $Z \sim N(0,1)$ is a standard normal random variable and $X = e^{\mu+\sigma Z}$. This means that $X \sim \text{Lognormal}(\mu, \sigma^2)$ and so its expected value is $e^{\mu+\sigma^2/2}$. In the following parts, implement a method to estimate this expected value using at least $10,000$ samples. Use $\mu = 0.1$ and $\sigma = 0.2$.

Part 1

Estimate the expected value directly by simulating X. Report your estimate along with a 95% confidence interval.

```
N = 10000
mu = 0.1
sig = 0.2


def box_muller(n=1):
    out = np.empty(n)

    m = (n + 1) // 2
    size = (2, m)
    lamda = 1 / 2

    u1, u2 = np.random.random(size)

    v = -np.log(u1) / lamda
    w = u2 * 2 * np.pi

    x = np.sqrt(v) * np.cos(w)
    y = np.sqrt(v) * np.sin(w)

    out[:m] = x
    out[-m:] = y

    return out


Z = box_muller(N)
X = np.exp(mu + sig * Z)

zdelta = 1.96
var = np.var(X)
error_1 = zdelta * np.sqrt(var / N)

print("Direct method: ")
print(f"Estimate: {np.mean(X):.5f}")
print(
```

```
    f"Confidence Interval: {np.round(np.mean(X) + np.array([-1, 1]) * error_1, 5)}")
print("Error bar size = {:.5f}".format(error_1))
```

```
Direct method:
Estimate: 1.12473
Confidence Interval: [1.12023 1.12923]
Error bar size = 0.00450
```

Part 2

Estimate the expected value using Z as a control variate. You can estimate the optimal value of c from your full sample (i.e., you don't have to calculate it by hand or do a separate pilot study). Report your estimate along with a 95% confidence interval. How much did this reduce the error from part 1?

```
c = -np.cov(X, Z)[0, 1] / np.var(Z)

X_c = X + c * (Z - 0)
var_c = np.var(X_c)
error_c = zdelta * np.sqrt(var_c / N)
print("Control variate: ")
print(f"Exact solution: {np.exp(mu + sig**2/2):.5f}")
print(f"Estimate: {np.mean(X_c):.5f}")
print(f"Estimated c: {c:.5f}")
print(f"Confidence Interval: {np.round(np.mean(X_c) + np.array([-1, 1]) * error_c, 5)}")
print("Error bar size = {:.5f}".format(error_c))
print("Error reduction = {:.5f}%".format((1 - error_c / error_1) * 100))
```

```
Control variate:
Exact solution: 1.12750
Estimate: 1.12791
Estimated c: -0.22518
Confidence Interval: [1.12728 1.12854]
Error bar size = 0.00063
Error reduction = 85.91827%
```

Problem 3

Consider an asset whose value S_t is governed by the SDE

$$\mathrm{d}S_t = \mu S_t \,\mathrm{d}t + \sigma(t) S_t \,\mathrm{d}W_t$$

where μ is a constant and

$$\sigma(t) = 0.2 + 0.1 \sin(t)$$

Use the parameters $\mu = 0.1$, $T = 5$ and $S_0 = 1$.

For each part, implement your method using at least 100 time steps and at least 10,000 samples.

```
mu = 0.1
T = 5
S0 = 1
t = np.linspace(0, T, 100)
dt = t[1] - t[0]
```

```
N = 100000
```

Part 1

Estimate $\mathbb{E}[S_T]$ using direct simulation. That is, use the Euler method to simulate S_t up until time T at least $10,000$ times and then average the final values. Report your estimate along with a 95% confidence interval.

```
sig = 0.2 + 0.1 * np.sin(t)
N = 10000

def sample_path():
    S = np.zeros_like(t)
    S[0] = S0
    for i in range(len(t)-1):
        dWt = np.sqrt(dt) * box_muller()
        dSt = mu * S[i] * dt + sig[i] * S[i] * dWt
        S[i+1] = S[i] + dSt
    return S

S_T = np.zeros(N)
for i in range(N):
    S_T[i] = sample_path()[-1]

zdelta = 1.96
var = np.var(S_T)
error_1 = zdelta * np.sqrt(var / N)

print("Direct method: ")
print(f"Estimate: {np.mean(S_T):.5f}")
print(f"Confidence Interval: {np.round(np.mean(S_T) + np.array([-1, 1]) * error_1, 5)}")
print("Error bar size = {:.5f}".format(error_1))
```

```
Direct method:
Estimate: 1.64988
Confidence Interval: [1.63263 1.66713]
Error bar size = 0.01725
```

Part 2

Let X_t be the value of a hypothetical asset governed by

$$\mathrm{d}X_t = \mu X_t \,\mathrm{d}t + \overline{\sigma} X_t \,\mathrm{d}W_t$$

where

$$\overline{\sigma} = \frac{1}{T}\int_0^T \sigma(t)\,\mathrm{d}t$$

Notice that $X_t \sim \mathrm{GBM}(\mu, \sigma)$ and so we know the distribution and expected value of X_T.

Estimate $\mathbb{E}[S_T]$ using X_T as a control variate. (This means that you should use the Euler method to simulate both S_t and X_t with *the same* noise and use the resulting X_T as a control). You can estimate the optimal value of from your full sample (i.e., you don't have to calculate it by hand or do a separate pilot study). Report your estimate along with a 95% confidence interval. How much did this reduce the error from part 1?

```
sig_bar = 1/T * (0.2 * T - 0.1 * np.cos(T) + 0.1)
def sample_path():
    S = np.zeros_like(t)
    X = np.zeros_like(t)
    S[0] = S0
    X[0] = S0

    for i in range(len(t)-1):
        dWt = np.sqrt(dt) * box_muller()
        dSt = mu * S[i] * dt + sig[i] * S[i] * dWt
        # dXt = mu * S[i] * dt + sig_bar * dWt
        dXt = mu * X[i] * dt + sig_bar * X[i] * dWt
        S[i+1] = S[i] + dSt
        X[i+1] = X[i] + dXt

    return S, X

S_T = np.zeros(N)
X_T = np.zeros(N)

for i in range(N):
    S, X = sample_path()
    S_T[i] = S[-1]
    X_T[i] = X[-1]

c = -np.cov(S_T, X_T)[0, 1] / np.var(X_T)
S_T_c = S_T + c * (X_T - S0 * np.exp(mu * T))
var_c = np.var(S_T_c)
error_c = zdelta * np.sqrt(var_c / N)
print("Control variate: ")
print(f"Estimate: {np.mean(S_T_c):.5f}")
print(f"Confidence Interval: {np.round(np.mean(S_T_c) + np.array([-1, 1]) * error_c, 5)}")
print(f"Estimated c: {c:.5f}")
print("Error bar size = {:.5f}".format(error_c))
print("Error reduction = {:.5f}%".format((1 - error_c / error_1) * 100))
```

```
Control variate:
Estimate: 1.64634
Confidence Interval: [1.64069 1.65199]
Estimated c: -0.99924
Error bar size = 0.00565
Error reduction = 67.22046%
```

Problem 4

Suppose that $X \sim \text{Exp}(1/2)$ and $Y \sim \text{Exp}(1/3)$ are independent random variables. Consider the probability

$$\mathbb{P}[X + Y > 4]$$

For each part, implement your method using at least $10,000$ samples.

Part 1

Estimate this probability directly by simulating both X and Y. Report your estimate along with a 95% confidence interval.

The estimate of the probability is

$$\theta = \mathbb{P}[X + Y > 4] = E[Z]$$

where $Z = \mathbb{1}_{\{X+Y>4\}}(X, Y)$

```
N = 10000
lam1 = 1/2
lam2 = 1/3
zdelta = 1.96

U1 = np.random.random(N)
U2 = np.random.random(N)

X = -np.log(U1) / lam1
Y = -np.log(U2) / lam2
Z = (X+Y) > 4
var = np.var(Z)
error = zdelta * np.sqrt(var / N)

print("Direct method: ")
print(f"Estimate: {np.mean(Z):.5f}")
print(f"Confidence Interval: {np.round(np.mean(Z) + np.array([-1, 1]) * error, 5)}")
print("Error bar size = {:.5f}".format(error))
```

```
Direct method:
Estimate: 0.51360
Confidence Interval: [0.5038 0.5234]
Error bar size = 0.00980
```

Part 2

Estimate this probability using conditional Monte Carlo. Choose whether to condition on X or Y and justify your choice. Report your estimate along with a 95% confidence interval. How much did this reduce the error from part 1?

Since $X \sim \text{Exp}(1/2)$ and $Y \sim \text{Exp}(1/3)$, the expectations are given by $E[X] = 2$ and $E[Y] = 3$

That is, Y has a relatively higher chance to determine the value of $Z = \mathbb{1}_{\{X+Y>4\}}$ because X, Y are both nonnegative and Y tends to have a bigger value than X.

Therefore, we can expect that conditioning on X to be more effective, since X will have the smaller covariance, and thus less dependent with Z than Y in a relative sense.

```
print(f"Cov(X, Z) = {np.cov(X, Z)[0, 1]:.4f}")
print(f"Cov(Y, Z) = {np.cov(Y, Z)[0, 1]:.4f}")
```

```
Cov(X, Z) = 0.4366
Cov(Y, Z) = 0.8522
```

Consider using conditional Monte Carlo by conditioning on X.

$$\begin{aligned}
E[Z|X = x] &= P[X + Y > 4|X = x] \\
&= P[Y > 4 - x] \\
&= 1 - P[Y < 4 - x] \\
&= 1 - F_Y(4 - x)
\end{aligned}$$

Note that cdf of Y is given by,

$$F_Y(y) = \begin{cases} 0 & \text{if } y < 0 \\ 1 - e^{-y/\lambda_Y} & \text{if } 0 \leq y \end{cases}$$
$$= \max\{1 - e^{-x/\lambda_Y}, 0\}$$

Note that the support of X is given by $x \geq 0$ and

$$\begin{aligned} 4 - x < 0 \quad & \text{if } x > 4 \\ 4 - x \geq 0 \quad & \text{if } x \leq 4 \end{aligned}$$

$$\begin{aligned} E[Z|X = x] &= 1 - F_Y(4 - x) \\ &= 1 - \max\left\{1 - \exp\left(\frac{-(4-x)}{\lambda_Y}\right), 0\right\} \\ &= 1 - \max\left\{1 - \exp\left(\frac{x-4}{\lambda_Y}\right), 0\right\} \\ &= \min\left\{\exp\left(\frac{x-4}{\lambda_Y}\right), 1\right\} \end{aligned}$$

```
Z_X = np.minimum(np.exp((X - 4) / 3), 1)

zdelta = 1.96
var_X = np.var(Z_X)
error_X = zdelta * np.sqrt(var_X / N)

print("Conditional Monte Carlo conditioning on X: ")
print(f"Estimate: {np.mean(Z_X):.5f}")
print(f"Confidence Interval: {np.round(np.mean(Z_X) + np.array([-1, 1]) * error_X, 5)}")
print("Error bar size = {:.5f}".format(error_X))
print("Error reduction from conditioning on X = {:.5f}%".format((1 - error_X / error) * 100))
```

```
Conditional Monte Carlo conditioning on X:
Estimate: 0.51822
Confidence Interval: [0.51338 0.52306]
Error bar size = 0.00484
Error reduction from conditioning on X = 50.60497%
```

We can verify that conditioning on X is more effective by calculating the error reduction from conditioning on Y. Notice that we can use the identical derivation of the conditional expection of Z as we did for conditioning on X. That is,

$$E[Z|Y = y] = \min\left\{\exp\left(\frac{y-4}{\lambda_X}\right), 1\right\}$$

```
Z_Y = np.minimum(np.exp((Y - 4) / 2), 1)
var_Y = np.var(Z_Y)
error_Y = zdelta * np.sqrt(var_Y / N)

print("Conditional Monte Carlo conditioning on Y: ")
print(f"Estimate: {np.mean(Z_Y):.5f}")
print(f"Confidence Interval: {np.round(np.mean(Z_Y) + np.array([-1, 1]) * error_Y, 5)}")
print("Error bar size = {:.5f}".format(error_Y))
print("Error reduction from conditioning on Y = {:.5f}%".format((1 - error_Y / error) * 100))
```

```
Conditional Monte Carlo conditioning on Y:
Estimate: 0.51590
Confidence Interval: [0.50926 0.52254]
Error bar size = 0.00664
Error reduction from conditioning on Y = 32.23308%
```

The result is consistent with the reasoning. The error reduction from conditioning on Y is smaller than the one from conditioning on X. Therefore, conditioning on X is more effective.