This is a **closed-book test**: no books, no notes, no calculators, no phones, no tablets, no computers (of any kind) allowed.

Do **NOT** turn this page over until you are **TOLD** to start.

Duration of the test: 3 hours.

Write your answers in the test booklets provided.

Please fill-in **ALL** the information requested on the front cover of **EACH** test booklet that you use.

The test consists of 5 pages, including this one. Make sure you have all 5 pages.

The test consists of 4 questions. **Answer all 4 questions**. The mark for each question is listed at the start of the question.

The test was written with the intention that you would have ample time to complete it. You will be rewarded for concise well-thought-out answers, rather than long rambling ones. **We seek quality rather than quantity**.

Moreover, an answer that contains relevant and correct information as well as irrelevant or incorrect information will be awarded fewer marks than one that contains the same relevant and correct information only.

Write legibly. Unreadable answers are worthless.

1. [5 marks]

Assume both x and y are positive real numbers and $x \approx y$, but $x \neq y$. In this case, we would expect some cancellation in computing $\log_e(x) - \log_e(y)$. On the other hand,

$$\log_e(x) - \log_e(y) = \log_e(x/y)$$

and $\log_e(x/y)$ involves no subtractions (hence no cancellation).

Would you expect the computed value of $\log_e(x/y)$ to be more accurate than the computed value of $\log_e(x) - \log_e(y)$?

Justify your answer.

2. [5 marks]

The Cauchy distribution with scale parameter $\sigma > 0$ (i.e., the Cauchy(σ) distribution) has pdf

$$f(x) = \frac{\sigma}{\pi(x^2 + \sigma^2)}$$

and CDF

$$F(x) = \frac{1}{2} + \frac{1}{\pi} \arctan\left(\frac{x}{\sigma}\right)$$

Assume that you have a pseudo-random-number generator, such as `rand` in MatLab, that generates independent pseudo-random Uniform $[0, 1]$ random variables. Discuss how you would generate a pseudo-random variable X with the Cauchy(σ) distribution.

3. [5 marks]

The method of *common random numbers* is a useful technique that we did not discuss in class. Suppose that g and h are closely related functions (i.e., $g(x) \approx h(x)$ for all x) and we want to find $\mathbb{E}[g(X) - h(X)]$, where the random variable X has CDF F (i.e., $X \sim F$). Note that

$$\mathbb{E}[g(X) - h(X)] = \mathbb{E}[g(X)] - \mathbb{E}[h(X)]$$

Suppose that we can generate pseudo random numbers $X \sim F$. Therefore, we could estimate $\mathbb{E}[g(X) - h(X)]$ from the Monte Carlo simulation

$$\mathrm{MC}_1 = \frac{1}{N} \sum_{n=1}^{N} \Big(g(X_n) - h(X_n) \Big)$$

where each $X_n \sim F$ and all the X_n, for $n = 1, 2, \ldots, N$, are independent. Note that MC_1 uses *common random numbers*. Alternatively, we could estimate $\mathbb{E}[g(X) - h(X)] = \mathbb{E}[g(X)] - \mathbb{E}[h(X)]$ from the Monte Carlo simulation

$$\mathrm{MC}_2 = \frac{1}{N} \sum_{n=1}^{N} g(X_{n,1}) - \frac{1}{N} \sum_{n=1}^{N} h(X_{n,2})$$

where each $X_{n,1} \sim F$, each $X_{n,2} \sim F$ and all the $X_{n,1}$ and $X_{n,2}$, for $n = 1, 2, \ldots, N$, are independent. MC_2 does not use *common random numbers*.

Which of these two Monte Carlo simulations (i.e., MC_1 or MC_2) is likely to be more efficient? Justify your answer.

4. [5 marks]

Consider an *exchange spread option* that is based on two underlyings, $S_t^{(1)}$ and $S_t^{(2)}$, with payoff at expiry (i.e., at time $t = T$) given by

$$h(S_T^{(1)}, S_T^{(2)}) = \max(S_T^{(1)} - S_T^{(2)} - \hat{K}, 0)$$

where $\hat{K}$ is a constant. Assume that the two underlyings, $S_t^{(1)}$ and $S_t^{(2)}$, start with values $S_0^{(1)}$ and $S_0^{(2)}$, respectively, at time $t = 0$ and evolve in time according to the SDEs

$$\begin{aligned} dS_t^{(1)} &= rS_t^{(1)}dt + \sigma_1 S_t^{(1)} dW_t^{(1)} \\ dS_t^{(2)} &= rS_t^{(2)}dt + \sigma_2 S_t^{(2)} dW_t^{(2)} \end{aligned}$$

where r is the risk free interest rate, σ_1 and σ_2 are the volatilities associated with $S_t^{(1)}$ and $S_t^{(2)}$, respectively, and the Brownian motions, $W_t^{(1)}$ and $W_t^{(2)}$, are correlated with correlation coefficient $\rho \in [-1, 1]$. Hence,

$$\begin{aligned} S_T^{(1)} &= S_0^{(1)} \mathrm{e}^{(r-\sigma_1^2/2)T + \sigma_1 W_T^{(1)}} \\ S_T^{(2)} &= S_0^{(2)} \mathrm{e}^{(r-\sigma_2^2/2)T + \sigma_2 W_T^{(2)}} \end{aligned}$$

and

$$\begin{aligned} W_T^{(1)} &= \sqrt{T}\Big(\sqrt{1-\rho^2} Z^{(1)} + \rho Z^{(2)}\Big) \\ W_T^{(2)} &= \sqrt{T} Z^{(2)} \end{aligned}$$

where $Z^{(1)} \sim N(0,1)$, $Z^{(2)} \sim N(0,1)$ and $Z^{(1)}$ and $Z^{(2)}$ are independent.

The price of this option at time $t = 0$ is

$$P_0 = \mathbb{E}[\mathrm{e}^{-rT} h(S_T^{(1)}, S_T^{(2)})]$$

We can easily do a "standard" Monte Carlo simulation to approximate P_0 as follows.

(a) For $n = 1, 2, \ldots, N$, let

$$Y_n = \mathrm{e}^{-rT} h(S_{T,n}^{(1)}, S_{T,n}^{(2)})$$

where

$$\begin{aligned} S_{T,n}^{(1)} &= S_0^{(1)} \mathrm{e}^{(r-\sigma_1^2/2)T + \sigma_1 W_{T,n}^{(1)}} \\ S_{T,n}^{(2)} &= S_0^{(2)} \mathrm{e}^{(r-\sigma_2^2/2)T + \sigma_2 W_{T,n}^{(2)}} \end{aligned}$$

and

$$\begin{aligned} W_{T,n}^{(1)} &= \sqrt{T}\Big(\sqrt{1-\rho^2} Z_n^{(1)} + \rho Z_n^{(2)}\Big) \\ W_{T,n}^{(2)} &= \sqrt{T} Z_n^{(2)} \end{aligned}$$

and each $Z_n^{(1)} \sim N(0,1)$, each $Z_n^{(2)} \sim N(0,1)$ and all the $Z_n^{(1)}$ and $Z_n^{(2)}$, for $n = 1, 2, \ldots, N$, are independent.

(b) Approximate the option price P_0 by

$$\hat{P}_0 = \frac{1}{N}\sum_{n=1}^{N} Y_n$$

Assume that you have a function, such as `blsprice` in MatLab,

`[Call, Put] = blsprice(`S_0`,` K`,` r`,` T`,` σ`)`

that computes the price at time $t = 0$ of a "vanilla" call or put option that expires at time $t = T$, with strike price K, risk free interest rate r, volatility σ and underlying S_t that starts with value S_0 at time $t = 0$ and evolves in time according to the SDE

$$dS_t = rS_t dt + \sigma S_t dW_t$$

Assume also that you have a program such as `randn` in MatLab that generates independent standard normal random numbers (i.e., generates independent $Z \sim N(0, 1)$).

How can you use `blsprice` and `randn` together with conditional expectation to develop a more efficient Monte Carlo simulation than the "standard' one given above to approximate the price P_0 of this exchange spread option?

The description of your new Monte Carlo simulation should be detailed enough so that someone who does not know any mathematical finance can implement it easily in MatLab.