

This is a **closed-book test**: no books, no notes, no calculators allowed.

1. [10 marks: 5 marks for each part]

In your calculus course, you probably learnt that

$$e = \lim_{n \rightarrow \infty} (1 + 1/n)^n$$

where $e \approx 2.718281828459$ is the base of the natural logarithms. (If you don't recall this, just assume that it is true for this question.)

I wrote a little MATLAB program that uses IEEE double-precision floating-point numbers and arithmetic to approximate e by $(1 + 1/n)^n$ for various finite values of n . The numerical results produced by the program are presented in the following table.

| n | $(1 + 1/n)^n$ | $e - (1 + 1/n)^n$ |
|-----------|-------------------|-------------------|
| 10^1 | 2.593742460100002 | 1.24539e-01 |
| 10^2 | 2.704813829421528 | 1.34679e-02 |
| 10^3 | 2.716923932235594 | 1.35789e-03 |
| 10^4 | 2.718145926824926 | 1.35901e-04 |
| 10^5 | 2.718268237192297 | 1.35912e-05 |
| 10^6 | 2.718280469095753 | 1.35936e-06 |
| 10^7 | 2.718281694132082 | 1.34326e-07 |
| 10^8 | 2.718281798347358 | 3.01116e-08 |
| 10^9 | 2.718282052011560 | -2.23552e-07 |
| 10^{10} | 2.718282053234788 | -2.24775e-07 |
| 10^{11} | 2.718282053357110 | -2.24898e-07 |
| 10^{12} | 2.718523496037238 | -2.41667e-04 |
| 10^{13} | 2.716110034086901 | 2.17179e-03 |
| 10^{14} | 2.716110034087023 | 2.17179e-03 |
| 10^{15} | 3.035035206549262 | -3.16753e-01 |
| 10^{16} | 1.000000000000000 | 1.71828e+00 |
| 10^{17} | 1.000000000000000 | 1.71828e+00 |
| 10^{18} | 1.000000000000000 | 1.71828e+00 |
| 10^{19} | 1.000000000000000 | 1.71828e+00 |
| 10^{20} | 1.000000000000000 | 1.71828e+00 |

- (a) Why does the program produce the very poor approximation 1 to e for $n = 10^{16}$, 10^{17} , 10^{18} , 10^{19} and 10^{20} ?
- (b) Why does the magnitude of the error reach a minimum of about $3 \cdot 10^{-8}$ at $n = 10^8$?

2. [5 marks]

Suppose you have a random number generator, such as MATLAB's `randn`, that computes independent standard normal random numbers (i.e., independent random numbers from the distribution $N(0, 1)$). Describe how you can generate a vector of multivariate normal random numbers $X \in \mathbb{R}^2$ having mean $\mu \in \mathbb{R}^2$ and covariance matrix $\Sigma \in \mathbb{R}^{2 \times 2}$ (i.e., $X \sim N(\mu, \Sigma)$), where

$$\mu = E(X) = \begin{pmatrix} -2 \\ 5 \end{pmatrix}$$

$$\Sigma = \text{var}((X - \mu)(X - \mu)^T) = \begin{pmatrix} 2.5 & 0.5 \\ 0.5 & 2.5 \end{pmatrix}$$

and v^T is the transpose of the column vector v .

3. [10 marks: 5 marks for each part]

In one of your assignments, you used stratified sampling to price a European *up-and-out* barrier call option with parameters

$$S_0 = 100, \quad K = 100, \quad B = 110, \quad r = 0.05 \quad \sigma = 0.2, \quad T = 0.25$$

which has the payoff $\max(S_T - K, 0)$ if $S_t < B$ for all $t \in [0, T]$ and payoff 0 if $S_t \geq B$ for any $t \in [0, T]$.

For this problem, consider using control variates to price this European *up-and-out* barrier call option. For the control, use a standard European call option with the same parameters as those given above for the European *up-and-out* barrier call option, except that the barrier value B is not relevant for the standard European call option.

- (a) Describe how you would write a MATLAB program that uses control variates to approximate the price of this European *up-and-out* barrier call option.
(You don't have to actually write a complete program. You just have to describe its overall structure, its essential components and its key parameters.)
- (b) Do you expect that the control variates approach described above will be an effective variance reduction technique for pricing this European *up-and-out* barrier call option?

Justify your answer.

4. [5 marks]

When we discussed stratified sampling in class, we discussed the following little example from your textbook.

Suppose you want to estimate

$$\theta = E(h(U)) = \int_0^1 h(u) du$$

where $U \sim U(0, 1)$ is a uniform random variable drawn from the interval $[0, 1]$.

Standard Monte Carlo estimates θ by

$$\tilde{\theta} = \frac{1}{N} \sum_{k=1}^N h(U_k)$$

where the $U_k \sim U(0, 1)$ are independent uniform random numbers drawn from the interval $[0, 1]$.

We can divide the interval $[0, 1]$ up into m subintervals (or strata) $[\frac{j-1}{m}, \frac{j}{m}]$, for $j = 1, 2, \dots, m$, all of the same length $\frac{1}{m}$. Thus, the probability that a uniform random number $U \sim U(0, 1)$ will lie in any one of these subintervals is $p = \frac{1}{m}$.

For each $j = 1, 2, \dots, m$, we condition on $U \in [\frac{j-1}{m}, \frac{j}{m}]$ and compute

$$\hat{\theta}_j = \frac{1}{N_j} \sum_{k=1}^{N_j} h\left(\frac{U_k + j - 1}{m}\right)$$

where $U_k \sim U(0, 1)$ are independent uniform random numbers drawn from the interval $[0, 1]$ and the N_j are positive integers satisfying

$$N = \sum_{j=1}^m N_j$$

Then we can estimate θ by

$$\hat{\theta} = \sum_{j=1}^m \frac{1}{m} \hat{\theta}_j$$

We also noted in class that

$$\text{var}(\hat{\theta}) = \sum_{j=1}^m \frac{1}{m^2} \text{var}(\hat{\theta}_j) = \sum_{j=1}^m \frac{1}{m^2} \frac{\text{var}(X_j)}{N_j}$$

where X_j is the random variable sampled in the j -th subinterval $[\frac{j-1}{m}, \frac{j}{m}]$.

To make the calculations below simpler, assume that $h(u) = u$ for all $u \in [0, 1]$ and that $N_j = N/m$ for all $j = 1, 2, \dots, m$, where m (the number of subintervals or strata) is an integer satisfying $1 \leq m \leq N$ and m divides N evenly.

For the special case described in the paragraph above, compute $\text{var}(\tilde{\theta})$ and $\text{var}(\hat{\theta})$.

How should you choose the parameter m to minimize your expression for $\text{var}(\hat{\theta})$ that you computed above? What does this imply about $N_j = N/m$ for all $j = 1, 2, \dots, m$?

How effective is stratified sampling as a variance reduction technique in this special case?