# Solution to the Midterm Test

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

   The students were asked to consider the expression

$$\sqrt{1+\sin^2(x)} - \cos(x) \qquad \text{for } x \in [-\tfrac{\pi}{4}, \tfrac{\pi}{4}] \tag{1}$$

   I told them that, when it is evaluated using IEEE Double-Precision Floating-Point Arithmetic, the computed value for expression (1) is inaccurate in a relative error sense for a range of values of $x$ within the interval $[-\frac{\pi}{4}, \frac{\pi}{4}]$, even though the computed values of both $\sin(x)$ and $\cos(x)$ are accurate for all $x \in [-\frac{\pi}{4}, \frac{\pi}{4}]$.

   (a) They are asked to give a specific value of $x \in [-\frac{\pi}{4}, \frac{\pi}{4}]$ for which the computed value of expression (1) has a very large relative error and to show that the relative error is orders of magnitude larger than the *machine epsilon* (i.e., $\epsilon_{\text{mach}}$) for IEEE Double-Precision Floating-Point Numbers.

   For $x \in [-\frac{\pi}{4}, \frac{\pi}{4}]$, I think the only values of $x$ for which (1) is very inaccurate are those for which $|x| \ll 1$, but $x \neq 0$. For such $x$, there is catastrophic cancellation when evaluating (1), since

$$\sqrt{1+\sin^2(x)} \approx 1 \qquad \text{and} \qquad \cos(x) \approx 1$$

   Note that I have excluded $x = 0$, since, for $x = 0$, both the exact and computed values of (1) are zero. So, even though there is cancellation in (1), the error is zero.

   One specific value of $x$ for which the relative error associated with evaluating (1) in floating-point arithmetic is orders of magnitude larger than the *machine epsilon* (i.e., $\epsilon_{\text{mach}}$) for IEEE Double-Precision Floating-Point Numbers is $x = 10^{-10}$. There are many other values of $x$ as well, but let me explain why $x = 10^{-10}$ is a good example. You can use your judgement when assessing whether the examples the students give are also good examples.

   Since

$$\sin(x) = x - \frac{x^3}{3!} + \cdots$$

   it follows that

$$\sin^2(x) = x^2 - \frac{x^4}{3} + \cdots$$

   So,

$$\sin^2(10^{-10}) \approx 10^{-20} \ll \epsilon_{\text{mach}} \approx 2.22 \cdot 10^{-16}$$

   Therefore, when evaluated in IEEE Double-Precision Floating-Point Arithmetic,

$$\text{fl}\left(1+\sin^2(10^{-10})\right) = 1$$

   Hence,

$$\text{fl}\left(\sqrt{1+\sin^2(10^{-10})}\right) = 1$$

Also note that

$$\cos(x) = 1 - \frac{x^2}{2} + \cdots$$

Therefore, when evaluated in IEEE Double-Precision Floating-Point Arithmetic,

$$\text{fl}\left(\cos(10^{-10})\right) = 1$$

Hence, when evaluated in IEEE Double-Precision Floating-Point Arithmetic,

$$\text{fl}\left(\sqrt{1+\sin^2(10^{-10})} - \cos(10^{-10})\right) = 0$$

On the other hand, in exact arithmetic,

$$\sqrt{1+\sin^2(10^{-10})} - \cos(10^{-10}) > 0$$

since

$$\sin(10^{-10}) > 0$$

implies

$$\sqrt{1+\sin^2(10^{-10})} > 1$$

whereas

$$\cos(10^{-10}) < 1$$

Therefore the relative error associated with evaluating (1) in IEEE Double-Precision Floating-Point Arithmetic is

$$\frac{\text{fl}\left(\sqrt{1+\sin^2(10^{-10})} - \cos(10^{-10})\right) - \left(\sqrt{1+\sin^2(10^{-10})} - \cos(10^{-10})\right)}{\sqrt{1+\sin^2(10^{-10})} - \cos(10^{-10})} = -1$$

That is, the relative error associated with evaluating (1) in IEEE Double-Precision Floating-Point Arithmetic is orders of magnitude larger than the *machine epsilon* (i.e., $\epsilon_{\text{mach}}$) for IEEE Double-Precision Floating-Point Numbers. (They could add that $\epsilon_{\text{mach}} = 2^{-52} \approx 2.22 \cdot 10^{-16}$, but this is not really necessary. All that is really needed is the implicit recognition that $\epsilon_{\text{mach}}$ is orders of magnitude smaller than 1.

(b) I asked the students to find another expression that is mathematically equivalent to (1) that is accurate in a relative error sense for all $x \in [-\frac{\pi}{4}, \frac{\pi}{4}]$.
I also asked them to explain why they believe the computed value of their new expression is accurate in a relative error sense for all $x \in [-\frac{\pi}{4}, \frac{\pi}{4}]$.

The goal here is to find another expression that is mathematically equivalent to (1), but does not suffer from catastrophic cancellation. One way to get such an expression is as follows.

$$\begin{aligned}
\sqrt{1+\sin^2(x)} - \cos(x) &= \left(\sqrt{1+\sin^2(x)} - \cos(x)\right) \frac{\sqrt{1+\sin^2(x)} + \cos(x)}{\sqrt{1+\sin^2(x)} + \cos(x)} \\
&= \frac{1+\sin^2(x) - \cos^2(x)}{\sqrt{1+\sin^2(x)} + \cos(x)} \\
&= \frac{2\sin^2(x)}{\sqrt{1+\sin^2(x)} + \cos(x)}
\end{aligned}$$

Therefore, the expression

$$\frac{2\sin^2(x)}{\sqrt{1+\sin^2(x)} + \cos(x)} \tag{2}$$

is mathematically equivalent to (1), but (2) does not suffer from catastrophic cancellation.

To explain why (2) is accurate in a relative error sense for all $x \in [-\frac{\pi}{4}, \frac{\pi}{4}]$ they could try to use a $\delta$-type rounding argument, but it's fairly long and complex. Since this part of the question is worth only two or three marks (depending on how many marks you decide to give for finding (2), I think it is fine if they use a "hand waiving" argument.

To start, note that I told them that the computed values of both $\sin(x)$ and $\cos(x)$ are accurate for all $x \in [-\frac{\pi}{4}, \frac{\pi}{4}]$. Therefore the numerator of (2) will be accurate, since it involves only the multiplication of accurate values. The term $\sqrt{1+\sin^2(x)}$ in the denominator will also be accurate because $\sin^2(x)$ will be accurate, the addition of two positive accurate values is accurate and the square root of an accurate value is accurate. One reason why I restricted $x$ to be in $[-\frac{\pi}{4}, \frac{\pi}{4}]$ is to ensure that $\cos(x)$ is accurate (in a relative error sense) and that $\cos(x) > 0$. Since then we can claim the denominator is accurate, because it is the sum of two positive accurate values. So, we have "hand waived" to the point that we can claim that both the numeration and denominator of (2) are accurate. However, the division of two accurate values is always accurate (assuming the denominator is not zero, which is the case here). So, the floating-point computation of (2) is accurate in a relative error sense.

2. [5 marks]

I started this question by reviewing the method I gave them in class that uses two Unif$[0,1]$ pseudo-random variables to generate one doubly-exponential pseudo-random variable, $X$.

In this question, I asked them to find a method that uses only one Unif$[0,1]$ pseudo-random variable (and no other pseudo-random variables) to generate a doubly-exponential pseudo-random variable, $X$.

During the test, I noticed that several students seemed to be trying to modify the method I gave them in class so that it uses only one Unif$[0,1]$ pseudo-random variable (and no other pseudo-random variables) to generate a doubly-exponential pseudo-random variable, $X$. I don't think this is possible. (Let me know if I am wrong about this.) However, they can easily develop a method based on the inverse transform method that meets the requirement.

Unfortunately, I gave them the wrong formula for the pdf of the doubly-exponential pseudo-random variable. Luckily, though, one of your students found my error and I think we both told our groups that the pdf of a doubly-exponential pseudo-random variable is

$$f(x) = \frac{\lambda}{2}\,\mathrm{e}^{-\lambda|x|} \qquad \text{for } x \in \mathbb{R}$$

From this it follows that the CDF of a doubly-exponential pseudo-random variable is

$$F(x) = \int_{-\infty}^{x} f(x)\,dx = \begin{cases} \frac{1}{2}\mathrm{e}^{\lambda x} & \text{for } x \le 0 \\ 1 - \frac{1}{2}\mathrm{e}^{-\lambda x} & \text{for } x > 0 \end{cases}$$

Moreover, note that $F(0) = \frac{1}{2}$.

For the inverse transform method, we generate a pseudo-random variable $U \sim$ Unif$[0,1]$ and then solve

$$F(X) = U \tag{3}$$

for a doubly-exponential pseudo-random variable, $X$. If $U \le \frac{1}{2}$, then (3) becomes

$$\frac{1}{2}\mathrm{e}^{\lambda X} = U$$

which gives

$$X = \frac{1}{\lambda}\log(2U)$$

On the other hand, if $U > \frac{1}{2}$, then (3) becomes

$$1 - \frac{1}{2}\mathrm{e}^{-\lambda X} = U$$

which gives

$$X = -\frac{1}{\lambda}\log(2(1-U))$$

Hence, the method is

Inverse transform method to compute a doubly-exponential pseudo-random variable, $X$.

(1) generate a pseudo-random variable $U \sim \text{Unif}[0, 1]$

(2) if $U \leq 1/2$
then $X = \frac{1}{\lambda} \log(2U)$
else $X = -\frac{1}{\lambda} \log(2(1 - U))$

3. [5 marks]

I told the students to assume $f(x) = \alpha \hat{f}(x)$, where $\hat{f}(x) \geq 0$ for all $x \in \mathbb{R}$ and $\alpha > 0$ is the normalizing constant so that $\int f(x)\,dx = 1$. That is, $f(x)$ is a probability density function. In many practical problems, you know $\hat{f}(x)$, but you don't know $\alpha$, and $\alpha$ is expensive to compute. However, you may know another probability density function $g(x)$, having a shape similar to $\hat{f}(x)$, for which $\hat{f}(x) \leq Mg(x)$ for all $x \in \mathbb{R}$ and it may be relatively easy to find such an $M$ and to generate a pseudo-random variable $Y$ with probability density function $g(x)$.

I also told them that they can make a small modification to the acceptance-rejection method that we discussed in class so that they can use this modified version of the acceptance-rejection method to generate a pseudo-random variable $X$ having probability density function $f(x)$ **without knowing** $\alpha$.

I asked them to explain how this can be done.

One of the two key observation needed to develop such a modified version of the acceptance-rejection method is that

$$f(x) = \alpha \hat{f}(x) \leq \alpha M g(x) \qquad \text{for all } x \in \mathbb{R}$$

which follows immediately from the assumptions above that

$$f(x) = \alpha \hat{f}(x) \qquad \text{for all } x \in \mathbb{R}$$

and

$$\hat{f}(x) \leq Mg(x) \qquad \text{for all } x \in \mathbb{R}$$

So, for $c = \alpha M$, we have

$$f(x) \leq cg(x) \qquad \text{for all } x \in \mathbb{R}$$

which is one of the requirements of the acceptance-rejection method that we talked about in class. Note that, since we don't know $\alpha$, we don't know $c$ explicitly, but we will see below that this doesn't matter.

The other key observation is that, using $f(x) = \alpha \hat{f}(x)$ and $c = \alpha M$, the acceptance condition

$$U \leq \frac{f(x)}{cg(x)} \tag{4}$$

for the acceptance-rejection method we discussed in class can be rewritten as

$$U \leq \frac{\alpha \hat{f}(x)}{\alpha M g(x)} = \frac{\hat{f}(x)}{Mg(x)} \tag{5}$$

So, we see from (5) that we can write the acceptance condition in a form that doesn't require us to know $\alpha$. We only need to know $\hat{f}(x)$, $M$ and $g(x)$, which is what I said to assume we know.

Hence, the modified acceptance-rejection method, that doesn't require us to know $\alpha$, is

Modified Acceptance-Rejection Method

(1) generate a $Y \sim g$

(2) generate a $U \sim \text{Unif}[0, 1]$

(3) if $U \leq \dfrac{\hat{f}(Y)}{Mg(Y)}$
then accept $X = Y$
else reject $Y$ and go to step (1)

4. [10 marks: 5 marks each part]

This question is about using Monte Carlo with Stratified Sampling to price a deep-out-of-the-money call option with underlying $S_t$ that satisfies the SDE

$$dS_t = rS_t dt + \sigma S_t dW_t \tag{6}$$

in the risk-neutral world, where $r$ is the risk-free interest rate and $\sigma$ is the volatility. The parameters for the option are:

- the initial stock price is $S_0 = \$80.00$,
- the strike price is $K = \$100.00$
- the time to maturity is $T = 0.25$ years,
- the risk-free interest rate is $r = 0.02$, and
- the volatility is $\sigma = 0.2$.

I told the students to use just two strata:

- stratum (1) corresponds to $S_T < K$, and
- stratum (2) corresponds to $S_T \geq K$.

I also told them that the $\hat{z}$ that satisfies the equation

$$K = S_0 \mathrm{e}^{(r - \frac{\sigma^2}{2})T + \sigma\sqrt{T}\hat{z}}$$

is

$$\hat{z} \approx 2.2314$$

and

$$\begin{aligned} p &= F(\hat{z}) \approx 0.98717 \\ q &= 1 - p \approx 0.012826 \end{aligned}$$

where $F$ is the CDF for the standard normal distribution (i.e., $N(0,1)$).

(a) I asked the students to describe how they could use Monte Carlo Stratified Sampling with the two strata described above to approximate efficiently the price of this deep-out-of-the-money call option. I also told them that, in describing their algorithm, they can assume they have access to functions that compute $F$, $F^{-1}$ (in MatLab these functions are `normcdf` and `norminv`, respectively) and a pseudo-random number generator (such as `rand` in MatLab) that computes uniform $[0,1]$ pseudo-random numbers.

I asked them to provide sufficient detail in the description of their Monte Carlo Stratified Sampling method so that an experienced MatLab programmer with no knowledge of computational finance can implement their method in MatLab.

The first thing to note is that, since $S_t$ satisfies the SDE (6), $S_T$ satisfies

$$S_T = S_0 \mathrm{e}^{(r-\frac{\sigma^2}{2})T+\sigma\sqrt{T}Z} \qquad \text{for } Z \sim N(0,1)$$

Also, I noted above that

$$K = S_0 \mathrm{e}^{(r-\frac{\sigma^2}{2})T+\sigma\sqrt{T}\hat{z}}$$

where

$$\hat{z} \approx 2.2314$$

Hence, stratum (1), which corresponds to $S_T < K$, also corresponds to $Z < \hat{z}$. Similarly, stratum (2), which corresponds to $S_T \geq K$, also corresponds to $Z \geq \hat{z}$. Therefore, we can use $Z < \hat{z}$ and $Z \geq \hat{z}$ to define the two strata. Moreover, the probability that $S_T$ is in stratum (1) is

$$P(S_T < K) = P(Z < \hat{z}) = F(\hat{z}) = p \approx 0.98717$$

and the probability that $S_T$ is in stratum (2) is

$$P(S_T \geq K) = P(Z \geq \hat{z}) = 1 - F(\hat{z}) = 1 - p = q \approx 0.012826$$

In our Monte Carlo Stratified Sampling method, there is no need to sample $S_T$ in stratum (1), since the payoff for a call option is zero for all $S_T$ in stratum (1). So, we only need to sample $S_T$ in stratum (2). To do this, we need to be able to generate $Z \sim N(0,1)$ conditional on $Z \geq \hat{z}$. We can generate such a $Z$ by

$$Z = F^{-1}(p + Uq), \qquad U \sim \text{Unif}[0,1]$$

where $F^{-1}$ is the inverse of the standard normal CDF function (i.e., `norminv` in MatLab). Therefore, if we know how many samples, $N$, of $S_T$ in stratum (2) we want to take in our Monte Carlo Stratified Sampling method, we can generate an approximation, $\theta_{\text{MCSS}}$, to the price of this deep-out-of-the-money call option by

$$\theta_{MCSS} = q\,\theta_2$$

where $\theta_2$ is Monte Carlo restricted to stratum (2):

$$\theta_2 = \frac{1}{N}\sum_{n=1}^{N} \mathrm{e}^{-rT}(S_T^{(n)} - K) \tag{7}$$

for

$$S_T^{(n)} = S_0 \mathrm{e}^{(r-\sigma^2/2)T+\sigma\sqrt{T}Z_n}$$

and

$$Z_n = F^{-1}(p + U_n q)$$

for

$$U_n \sim \text{Unif}[0,1]$$

Note that I used $S_T^{(n)} - K$ in (7), rather than the usual $\max(S_T^{(n)} - K, 0)$, since we know $S_T^{(n)} \geq K$ in stratum (2). The students don't have to do this, but it is a good idea if they do.

Therefore, assuming that $S_0$, $K$, $T$, $r$ and $\sigma$ are already initialized, we can write the Monte Carlo Stratified Sampling method in MatLab as follows.

```
zhat = ( log(K/S0) - (r - sigma^2/2)*T ) / ( sigma * sqrt(T) )
p = normcdf(zhat)
q = 1 - p
U = rand(N,1)
Z = norminv(p + U * q)
ST = S0 * exp( (r - sigma^2/2) * T + sigma * sqrt(T) * Z )
theta2 = exp(-r*T) * mean( ST - K )
thetaMCSS = q * theta2
```

Note that the students don't have to write a MatLab program as I have done above. I asked them only to provide sufficient detail in the description of their Monte Carlo Stratified Sampling method so that an experienced MatLab programmer with no knowledge of computational finance can implement their method in MatLab. However, if they do write a MatLab program or a MatLab pseudo-code, that's good.

David: when I ran a little simulation based on the MatLab code above, I got an approximate value for the option that was a little too high. The price I got was usually around 0.0459, while the price from `blkprice` was 0.0397. Do you see anything that might be wrong with what I wrote above?

(b) I asked the students if they expect that their method described in part (a) will be significantly more efficient than simple Monte Carlo for pricing this deep-out-of-the-money call option.

I told them to be as quantitative as possible in answering this question.

This question is essentially asking them to compare the variances for

- the Monte Carlo method with Stratified Sampling developed in part (a), and
- the simple Monte Carlo method.

From the note on Monte Carlo with Stratified Sampling that I gave them recently, it follows that

$$\mathrm{Var}(\theta_{MCSS}) = \frac{1}{N}\, q^2\, \mathrm{Var}(X_1)$$

where

$$X_1 = S_0 \mathrm{e}^{(r-\sigma^2/2)T + \sigma\sqrt{T} Z_1}$$

and

$$Z_1 \sim N(0,1) \quad | \quad Z \geq \hat{z}$$

On the other hand, the variance for simple Monte Carlo is

$$\mathrm{Var}(\theta) = \frac{1}{N}\,\mathrm{Var}(X)$$

where

$$X = S_0 \mathrm{e}^{(r-\sigma^2/2)T+\sigma\sqrt{T}Z}$$

and

$$Z \sim N(0,1)$$

So, the variance reduction is given by

$$\frac{\mathrm{Var}(\theta_{MCSS})}{\mathrm{Var}(\theta)} = q^2\,\frac{\mathrm{Var}(X_1)}{\mathrm{Var}(X)} \tag{8}$$

When I wrote this question, I thought that $\mathrm{Var}(X_1)$ and $\mathrm{Var}(X)$ would be about the same size. However, I just did a quick simulation and it seems that

$$\frac{\mathrm{Var}(X_1)}{\mathrm{Var}(X)} \approx 37$$

(at least for the few simple simulations that I ran). Fortunately, though $q^2 \approx 1.645 \cdot 10^{-4}$. So, for my simple simulations, I get

$$\frac{\mathrm{Var}(\theta_{MCSS})}{\mathrm{Var}(\theta)} = q^2\,\frac{\mathrm{Var}(X_1)}{\mathrm{Var}(X)} \approx 0.0061$$

This is not as good a variance reduction as I was hoping for (I thought it would be about $q^2$), but it is still fairly good.

Of course, the students won't be able to say anything about the values of $\mathrm{Var}(\theta_{MCSS})$ and $\mathrm{Var}(\theta)$. However, they should be able to derive a formula similar to (8) for the variance reduction and argue from the fact that $q \approx 10^{-4}$ that we should get a fairly good variance reduction.