Exercise 1 (6.1)

Data

```
CRNs with the (a + c) synchronization
delta = 10.0 avg = 55.84410000000003 var = 2113.079303120312
delta = 1.0 avg = 6.254299999999966 var = 38.38847035703572
delta = 0.1 avg = 0.643300000000003 var = 2.013266436643664

CRNs without synchronization
delta = 10.0 avg = 55.35869999999988 var = 3673.855819891997
delta = 1.0 avg = 5.370500000000023 var = 1902.0580355535608
delta = 0.1 avg = 0.011500000000000265 var = 1946.5368214321438

IRNs (with Bi set to 1.0)
delta = 10.0 avg = 47.285200000000145 var = 2859.504211381143
delta = 1.0 avg = 5.932600000000001 var = 2941.1475719972022
delta = 0.1 avg = 1.0009000000000035 var = 2934.330532243232

IRNs
delta = 10.0 avg = 61.31770000000015 var = 56509.30329704001
delta = 1.0 avg = 8.670700000000007 var = 46397.589820491965
delta = 0.1 avg = 2.9493000000000125 var = 44282.17194670484
```

Conclusion

We want to compare the 4 types of synchronization tried.

First of all, as expected, the CRN (a+c) strategy worked best. This is the same conclusion as the one reached in the book (so it's also kind of unexpected, these simulations never seem to work out as expected).

Contrary to the result from the book, the variance of the CRN without synchronization didn't get so low for the small values of δ (for $\delta = 0.1$, the value from the book was 726, much smaller than 1946). This suggest that some implementation detail was different (not sure which one though).

The IRNs worked as expected, that is to say the variance was huge when B_i isn't fixed, as in the empirical observations in the book. When $B_i = 1.0$ for all B_i, the variance was much smaller, and this makes a lot of sense since one of the sources of variablility has been disabled and we get its average. This practice (substituting a source of variability for the value of its mean) could make sense in some scenarios, depending on what parameter is observed, but would be horrible for example if we want to estimate some failure rate where the variability could provoke some system-wide changes).

As expected, as $\delta \to 0$, the difference gets harder to estimate and the noisier estimators (particularly IRNs) become pretty useless. The CRN (a+c) tend to keep up much better as it is supposed to do.

This is unfortunate, because the IRN simulations are much easier to implement...

Exercise 2 (6.2)

Data

```
Variance reduction for delta = 10.0
Mean with propor. allocation + CV       =    55.650
Variance with propor. allocation + CV   =   207.017

Variance reduction for delta = 1.0
Mean with propor. allocation + CV       =     6.270
Variance with propor. allocation + CV   =    15.704

Variance reduction for delta = 0.1
Mean with propor. allocation + CV       =     0.638
Variance with propor. allocation + CV   =     1.607
```

Conclusion

So I used the (a + c) synchronization for the CRNs, the same as for exercise 1.

If we check the results for exercise 1:

```
CRNs with the (a + c) synchronization
delta = 10.0 avg = 55.84410000000003 var = 2113.079303120312
delta = 1.0 avg = 6.254299999999966 var = 38.38847035703572
delta = 0.1 avg = 0.643300000000003 var = 2.013266436643664
```

We can see that the variance was reduced quite a bit compared to exercise 1:

$\delta = 10$: $2113.079 \rightarrow 207.017$

$\delta = 1$: $38.388 \rightarrow 15.704$

$\delta = 0.1$: $2.013 \rightarrow 1.607$

The ratio grows bigger as δ grows so the control variable helps a lot when the difference between the two systems gets bigger (although at some point, I expect it to make not much of a difference if δ gets too big, because the estimate would be pretty useless for both I guess).

If delta is small (0.1), the extra work might not make much sense, but for bigger values, it sure does.

This is a sad conclusion, since programming these things tends to give me headaches.

Exercise 4 (6.10)

Data

```
==== CRUDE ESTIMATOR RESULTS (0.0) ====================
beta =  1.0
meanY = 12.270398915786515,   Var(Y) = 246.2293125684911
meanX = 12.270398915786515,   Var(X) = 246.2293125684911
meanD = 0.0,   Var(D) = -3.410605131648481E-13
Covar(Y,X) = 246.22931256849128
Covar(Y,D) = -1.7053025658242404E-13

==== CV RESULTS (0.0) ====================
meanX = 12.105832683237722
Var(D) = -3.410605131648481E-13
ratio = -1.3851336772504644E-15



==== CRUDE ESTIMATOR RESULTS (75.0) ====================
beta =  1.0
meanY = 12.270398915786515,   Var(Y) = 246.2293125684911
meanX = 12.268266902320324,   Var(X) = 246.25836124721684
meanD = 0.002132013466191296,   Var(D) = 0.023268773503446027
Covar(Y,X) = 246.23220252110224
Covar(Y,D) = -0.0028899526111274554

==== CV RESULTS (75.0) ====================
meanX = 12.105832683237722
Var(D) = 0.023268773503446027
ratio = 9.448927291482577E-5



==== CRUDE ESTIMATOR RESULTS (80.0) ====================
beta =  0.9981809412635513
meanY = 12.270398915786515,   Var(Y) = 246.2293125684911
meanX = 12.243017025720734,   Var(X) = 246.48186099968055
meanD = 0.027381890065781178,   Var(D) = 0.41799251544807703
Covar(Y,X) = 246.1465905263618
Covar(Y,D) = 0.0827220421293191

==== CV RESULTS (80.0) ====================
meanX = 12.106082229614419
Var(D) = 0.41799251544807703
ratio = 0.0016958347918698113
```

```
==== CRUDE ESTIMATOR RESULTS (90.0) ====================
beta =  0.9609114394457439
meanY = 12.270398915786515,   Var(Y) = 246.2293125684911
meanX = 11.448613223222285,   Var(X) = 249.3080430191164
meanD = 0.8217856925642302,   Var(D) = 15.739764491961694
Covar(Y,X) = 239.8987955478229
Covar(Y,D) = 6.330517020668196

==== CV RESULTS (90.0) ====================
meanX = 12.080142920577472
Var(D) = 15.739764491961694
ratio = 0.06313380146646452



==== CRUDE ESTIMATOR RESULTS (95.0) ====================
beta =  0.8838385072189944
meanY = 12.270398915786515,   Var(Y) = 246.2293125684911
meanX = 9.73740547053013,   Var(X) = 245.7388981035333
meanD = 2.532993445256386,   Var(D) = 49.82491637645887
Covar(Y,X) = 221.07164714778276
Covar(Y,D) = 25.157665420708355

==== CV RESULTS (95.0) ====================
meanX = 11.830712642666452
Var(D) = 49.82491637645887
ratio = 0.202755513111591
```

Conclusion

Analysis of the results

The barrier value is given in the title of each section.

Note that CRNs were used betwen the simulations.

So, as expected, the VC reduction of the variance is very impressive. In the worst cases (e.g. at a barrier of 95) it still is able to reduce the variance immensely. The best values are obviously obtained when there is no difference between the two. I did some additionnal simulations with barriers set to 0 and 75. From this, we can observe that at 0, as expected, there is no difference between the two systems and this allows us to get $\beta = 1.0$ which in turn means that the estimator is then perfect (yields exactly the value expected by the Black-Scholes equation).

The value of β goes down slowly and in the worst case, it still is at 0.88 (for the barrier of 95).

So, as far as I know, this worked pretty well.

Other ways to improve the variance

The question also asks about other ways to reduce the variance of the simulation. According to the article referenced in the book (Monte Carlo methods for security pricing, Boyle, Broadie, and Glasserman, 1997), there are a few other ways to achieve this.

1- antithetic variates

The first one (apart from just a better estimator) was the antithetic variates. These can be used when the path of the GBM is generated, basically generating the opposite path. This assures that the sample mean is of 0 in the standard pricing scheme and also a reduced covariance without twice the computing cost (we get twice the sample size anyways which is pretty good already). The covariance induced between the pairs of variables generated is demonstrated to always be ≤ 0.

2- control variates (other ones)

Although I believe that the actual control variate is pretty much the best we could hope for, we could use another one that is also correlated and for which we know the expectation.

3- others

Well, the other ones are pretty complicated to understand (or even understand), so I'll just name them:

- Moment matching methods (quadratic resampling)
- Stratified and Latin hypercube sampling (this one is pretty nice)
- Importance sampling
- Conditional Monte Carlo
- Low-discrepency sequences

These are all the ones suggested in the article from Boyle. There are probably many other ways to do it.

Exercise 5 (6.16)

Data

```
========================================================================
VALIDATION

Value expected without barrier 18.656769422297188


REPORT on Tally stat. collector ==> Value without barrier (validation)
    num. obs.      min          max        average     standard dev.
    10000       18.657       18.657       18.657        0.000


REPORT on Tally stat. collector ==> Value without barrier (validation)
    num. obs.      min          max        average     standard dev.
    10000        0.000      118.004       18.663       17.952

========================================================================
BARRIERS

REPORT on Tally stat. collector ==> barrier = 0.0
    num. obs.      min          max        average     standard dev.
    10000        0.000        0.000        0.000        0.000

REPORT on Tally stat. collector ==> barrier = 0.0 with CMC
    num. obs.      min          max        average     standard dev.
    10000        0.000        0.000        0.000        0.000



REPORT on Tally stat. collector ==> barrier = 75.0
    num. obs.      min          max        average     standard dev.
    10000        0.000       15.080        0.014        0.311

REPORT on Tally stat. collector ==> barrier = 75.0 with CMC
    num. obs.      min          max        average     standard dev.
    10000        0.000        2.128        0.020        0.131



REPORT on Tally stat. collector ==> barrier = 80.0
    num. obs.      min          max        average     standard dev.
    10000        0.000       34.577        0.104        1.138
```

```
REPORT on Tally stat. collector ==> barrier = 80.0 with CMC
    num. obs.      min          max        average     standard dev.
    10000        0.000        4.438        0.130        0.495



REPORT on Tally stat. collector ==> barrier = 90.0
    num. obs.      min          max        average     standard dev.
    10000        0.000       56.555        2.031        6.006

REPORT on Tally stat. collector ==> barrier = 90.0 with CMC
    num. obs.      min          max        average     standard dev.
    10000        0.000       10.152        2.116        3.133



REPORT on Tally stat. collector ==> barrier = 95.0
    num. obs.      min          max        average     standard dev.
    10000        0.000       79.248        5.172        9.963

REPORT on Tally stat. collector ==> barrier = 95.0 with CMC
    num. obs.      min          max        average     standard dev.
    10000        0.000       13.812        5.361        5.203
```

I used the equation as described by Okten *et al.* (On pricing discrete barrier options using condigional expectation and importance sampling in Monte Carlo, Okten G, Salta E, Goncu A., 2008), end of p.485.

Basically, the value is

$$X_e = \begin{cases} BSM(S(t_k), t_k, T), & \text{if the barrier is crossed at time } t_k \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

the idea is that we know the expectation of the European call option, so if the barrier is crossed (becomes an Europen call option) we calculate directly the exact value of the option's expected payoff on the remaining trajectory instead of continuing.

It still is Monte Carlo but in the event of s_0 < barrier, then it is exactly the same as calculating by BSM (Black-Scholes).

As can be seen in the data, the improvement is much better when the barrier is hit early and frequently (as said, in the extreme case of starting under the barrier, it is equivalent to direct calculation by BSM).

So the experiments match the expectation (the CMC is always a better estimator).