Financial Engineering HW3

Group 2: Natalie Frantsits, Lara Hofmann, Eldar Kodzov, Chiara Lesa, Vedad Sehanovic

April 2022

Contents

Exercise 1

a)

The Vasicek model implies short rate dynamics following an Ornstein-Uhlenbeck process. It is a Gaussian and mean-reverting process, with parameter γ characterizing the speed of mean reversion. The model is characterized by a stochastic differential equation given by

$$dr_t = \gamma(\bar{r} - r_t)dt + \sigma dW_t$$

The simplified Euler discretization scheme for simulation of Vasicek model-based short rate paths with a fixed set of time points $0 = t_0 < t_1 < ... < t_m = T$ is given by eq. 3.45 (Glasserman):

$$r_{t_{i+1}} = r_{t_i} + \gamma(\bar{r} - r_{t_i})(t_{i+1} - t_i) + \sigma\sqrt{t_{i+1} - t_i}Z_{i+1}, \quad i = 0, 1, ..., m-1,$$

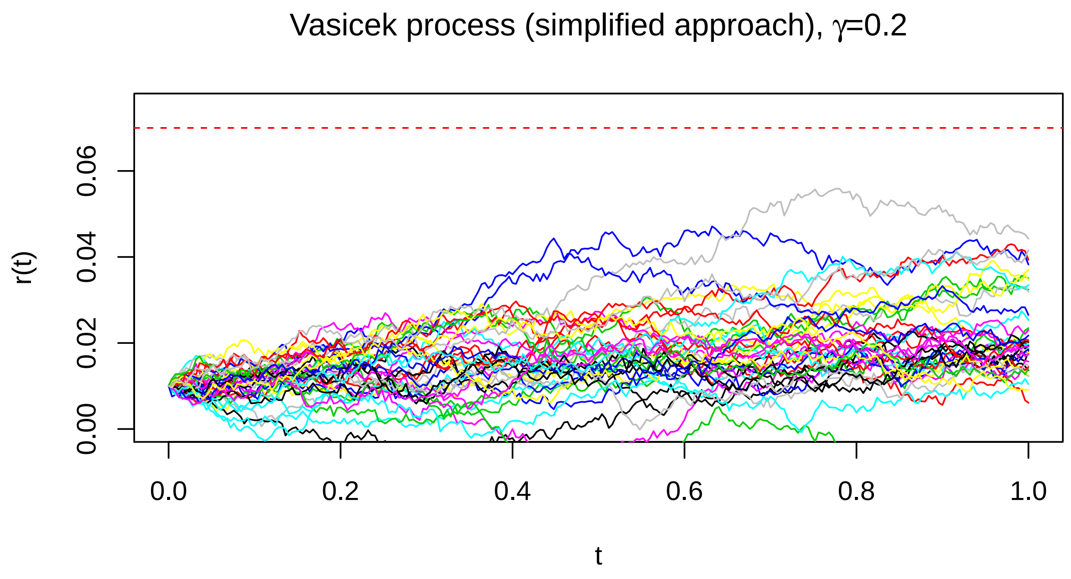
with iid standard normal rvs Z_i, $i = 1, ..., m$. Considering that this formula comes with a discretization error, and that $\bar{r}$ is constant, the exact simulation given by eq. 3.46 (Glasserman) can also be implemented:

$$r_{t_{i+1}} = r_{t_i} * e^{-\gamma(t_{i+1}-t_i)} + \bar{r}(1 - e^{-\gamma(t_{i+1}-t_i)}) + \sigma\sqrt{\frac{1}{2\gamma}(1 - e^{-2\gamma(t_{i+1}-t_i)})}Z_{i+1}$$

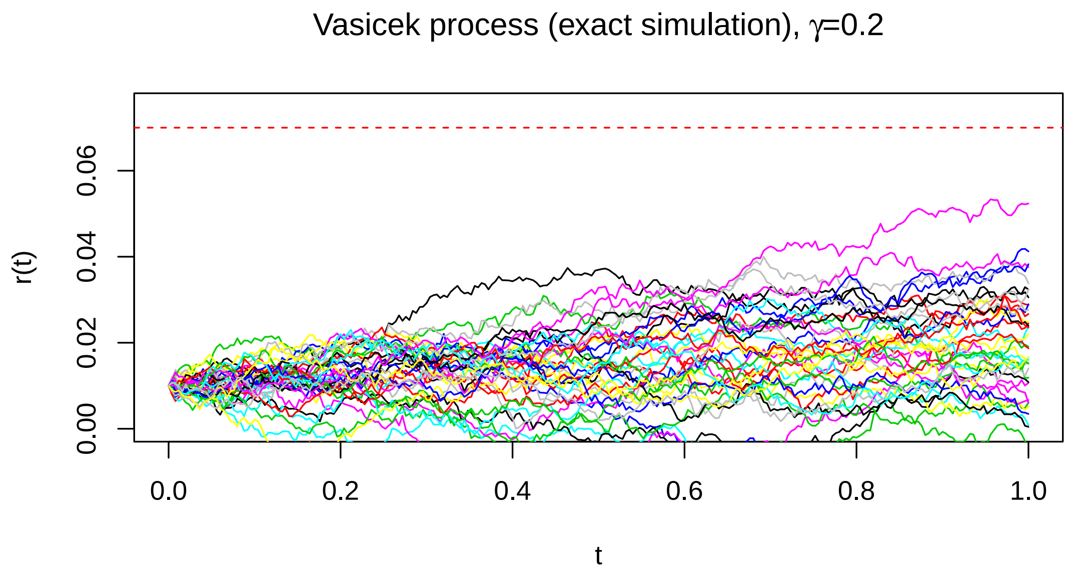
With 250 steps in time T, and constant step size, it is $t_i - t_{i-1} = 1/250$, $i = 1, ..., n$. Euler scheme for the two approaches can be implemented as follows:

```r
# gamma=0.2

r0 <- 0.01; r_bar <- 0.07; gamma <- 0.2; sigma <- 0.015
N <- 40
T <- 1
h <- 250
dt <- T/h

# Simplified approach eq 3.45.
r1 <- matrix(0,h+1,N)  # matrix to hold short rate paths
r1[1,] <- r0
set.seed(100)

for(j in 1:N){
  for(i in 2:(h+1)){
      dr <- gamma*(r_bar-r1[i-1,j])*dt + sqrt(dt)*sigma*rnorm(1,0,1)
      r1[i,j] <- r1[i-1,j] + dr
  }
}

# Non-simplified approach eq 3.46.
r1_nsa <- matrix(0,h+1,N)  # matrix to hold short rate paths
r1_nsa[1,] <- r0

for(j in 1:N){
  for(i in 2:(h+1)){
      r1_nsa[i,j] <- r1_nsa[i-1,j]*exp(dt*(-gamma)) + r_bar*(1-exp(-gamma*dt)) +
        sigma*sqrt((1-exp(-2*gamma*dt))/(2*gamma))*rnorm(1,0,1)
  }
}
```

```
# plot
t <- seq(0, T, dt)
  # simplified
matplot(t, r1[,1:40], type="l", lty=1, main = expression(paste("Vasicek process (simplified approach
abline(h=r_bar, col="red", lty=2)
```

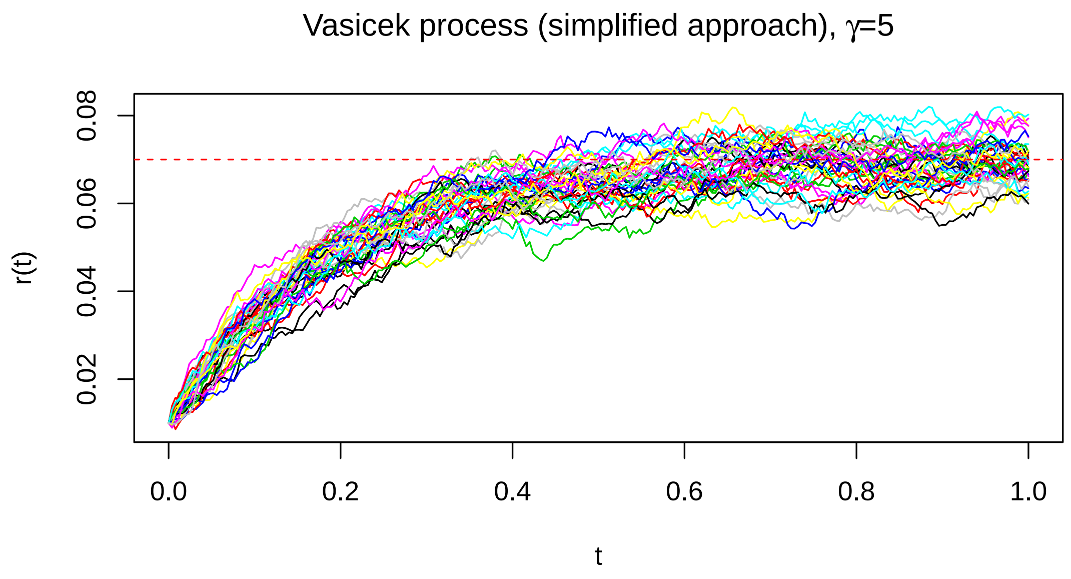


```
  # non-simplified
matplot(t, r1_nsa[,1:40], type="l", lty=1, main = expression(paste("Vasicek process (exact simulatio
abline(h=r_bar, col="red", lty=2)
```

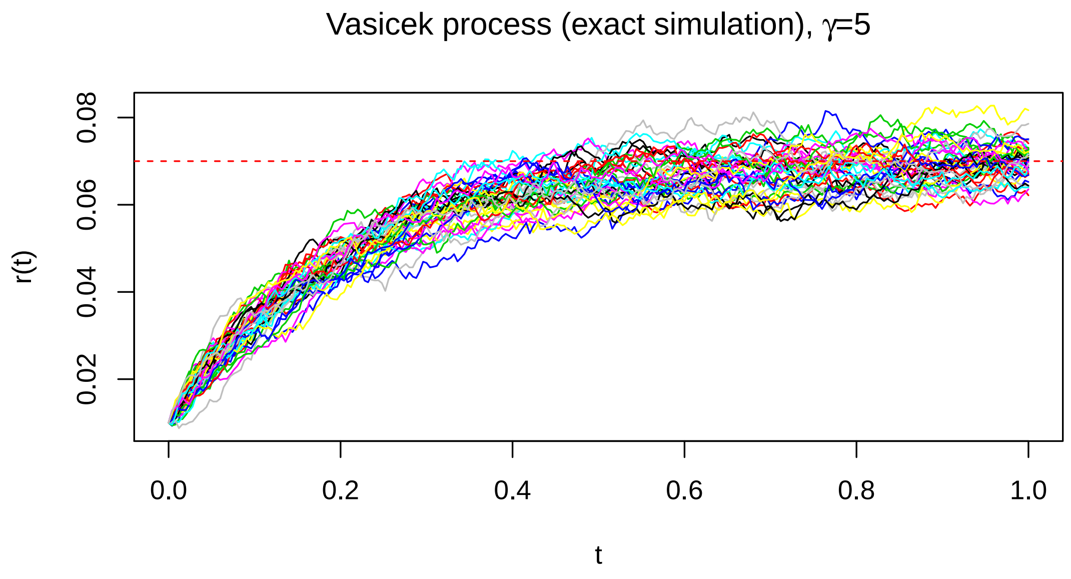


For $\gamma = 5$, the plots look distinctly different:

```
gamma <- 5

# Simplified approach eq 3.45.
r1_v20 <- matrix(0,h+1,N)  # matrix to hold short rate paths
r1_v20[1,] <- r0

for(j in 1:N){
  for(i in 2:(h+1)){
      dr <- gamma*(r_bar-r1_v20[i-1,j])*dt + sqrt(dt)*sigma*rnorm(1,0,1)
      r1_v20[i,j] <- r1_v20[i-1,j] + dr
  }
}

# Non-simplified approach eq 3.46.
r1_v21 <- matrix(0,h+1,N)  # matrix to hold short rate paths
r1_v21[1,] <- r0

for(j in 1:N){
  for(i in 2:(h+1)){
      r1_v21[i,j] <- r1_v21[i-1,j]*exp(dt*(-gamma)) + r_bar*(1-exp(-gamma*dt)) +
        sigma*sqrt((1-exp(-2*gamma*dt))/(2*gamma))*rnorm(1,0,1)
  }
}


# plot
t <- seq(0, T, dt)
  # simplified
matplot(t, r1_v20[,1:40], type="l", lty=1, main = expression(paste("Vasicek process (simplified appr
abline(h=r_bar, col="red", lty=2)
```



```
  # non-simplified
matplot(t, r1_v21[,1:40], type="l", lty=1, main = expression(paste("Vasicek process (exact simulatio
abline(h=r_bar, col="red", lty=2)
```



As expected, with higher speed of mean reversion γ, the paths converge to the long-run mean $\bar{r} = 0.05$ faster.

b)

The Cox-Ingersoll-Ross (CIR) model is similar to the Vasicek model, with only a slight change in the volatility term that ensures that no negative interest rates are possible for any parameter values, while preserving the mean-reverting property:

$$dr_t = \gamma(\bar{r} - r_t)dt + \sqrt{\alpha r_t} dW_t$$

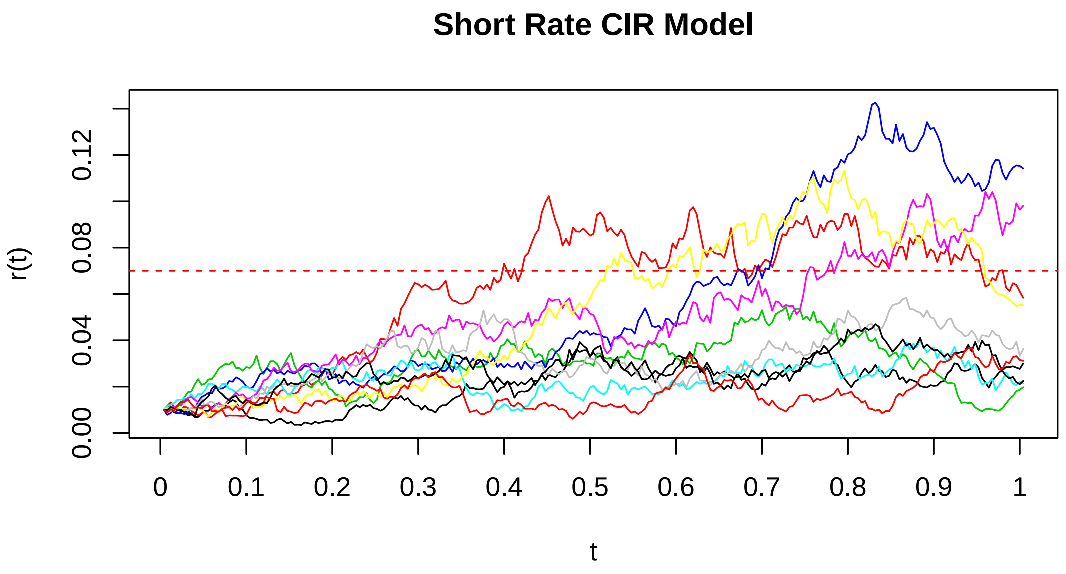
Euler discretization for this process can be implemented in line with eq. 3.66 (Glasserman) - for a fixed set of time points $0 = t_0 < t_1 < ... < t_m = T$, CIR short rate paths can be simulated as:

$$r_{t_{i+1}} = r_{t_i} + \gamma(\bar{r} - r_{t_i})(t_{i+1} - t_i) + \sqrt{\alpha r_{t_i}(t_{i+1} - t_i)} Z_{i+1}, \quad i = 0, 1, ..., m-1,$$

with iid standard normal rvs Z_i, $i = 1, ..., m$.
While CIR theoretically ensures that no negative interest rates occur, the discretization scheme does not guarantee non-negativity. The scheme is thus truncated as proposed on slide 117.

```
CIR <- function(N, r_0, r_bar, gamma, alpha, n) {
  path <- function() {
    dt <- 1/n
    r_t <- c(r_0, numeric(n))
    for (i in 1:n) {
      r_t[i+1] <- max(r_t[i] + gamma*(r_bar-r_t[i])*dt + sqrt(alpha*r_t[i]*dt)*rnorm(1), 0)
    }
    r_t
  }
```

```r
  replicate(N, path())
}

set.seed(10)
CIR_store <- CIR(10, 0.01, 0.07, 1.2, 0.04, 250)
```



Exercise 2

```r
euler.lookback <- function(n=250, T_=2, S_0=100, r=0.052,
                           V_0=0.16, V_bar=0.16, alpha=0.04,
                           xi=2, rho=-0.8, number_of_paths=10000){
  delta_t <- 1/n
  m <- T_/delta_t
  S <- matrix(numeric(0), nrow = number_of_paths, ncol = (m+1))
  V <- matrix(numeric(0), nrow = number_of_paths, ncol = (m+1))
  C <- numeric(0)
  SMin <- numeric(0)
  opt_monitor_freq=1/12
  # for the monitoring of the option
  # since the monitoring is done every month, only those S
  # will be observed out of the whole S time series
  # in 2 years 24 monitoring actions will be performed, which means 1 monitoring
  # every ( 250 * 2) / 24  = 21 days (approx)
  delta_monitoring <- opt_monitor_freq / T_
  yearly_steps_monitoring = ceiling(m  * delta_monitoring)
  #keep only one observation every month instead of daily
  freq_monitoring <- seq(0, m , by = yearly_steps_monitoring)
  for (i in 1:number_of_paths) {
    Z <- rnorm(m)
    Z_1 <- rnorm(m)
    Z_s <- rho*Z+sqrt(1-rho^2)*Z_1
    S[i,1] <- S_0
    V[i,1] <- V_0
    for (j in 1:m) {
      V[i, j+1] <- max(0, V[i,j]+alpha*delta_t*(V_bar-V[i,j])+xi*sqrt(V[i,j]*delta_t)*Z[j])
      S[i, j+1] <- max(0, S[i,j]*(1+r*delta_t)+S[i,j]*sqrt(V[i,j]*delta_t)*Z_s[j])
    }
  SMin[i]<- min(S[i, freq_monitoring])
  C[i] <- exp(-r*T_)*(S[i,(m+1)]-SMin[i])
  }
  mean(C)

}

euler.lookback()

## [1] 27.33746
```

Exercise 3

a) Suppose we have the setting given in page 135 of the slide set. Price this Asian option by using control variates method where you choose the standard European call option as the control variate ($Y_1 = e^{rT}(S_T K)^+$). Provide the corresponding 95% confidence interval.

b) Now take the geometric-average Asian option as control variate ($Y_2 = (A_{dg} K, 0)^+$) and compute the price of the arithmetic-average Asian option again. Provide the corresponding 95% confidence interval and compare this with the result you have obtained above. Which control variate results in more variance reduction? Justify your answer.

The following code computes the price of the geometric-average Asian option using the three methods: naive Monte-Carlo, control variates with a std. European call as control variate and control variates with a geometric-average Asian call as control variate. Furthermore, the 95% confidence intervals are computed.

```r
################### ASIAN CALL PRICING - NAIVE MONTE CARLO #####################

AsianCall_MC <- function(K,S0,sigma,r,T,delta,n){
  m <- T/delta
  C <- rep(0,times=n)
  for(i in 1:n){
    S <- rep(0,times=m+1)
    S[1] <- S0
    for(j in 1:m){
      S[j+1] <- S[j]*exp((r-sigma^2/2)*delta + sigma*sqrt(delta)*rnorm(1))
    }
    C[i] <- exp(-r*T)*max(mean(S[-1])-K,0)
  }
  C_hat <- mean(C)
  output <- list(C,C_hat)
  return(output)
}

################### EXACT BLACK SCHOLES EUROPEAN CALL PRICE ####################

BS_Call <- function(K,S0,sigma,r,T){
  d1 <- (log(S0/K) + (r+sigma^2/2)*T)/(sigma*sqrt(T))
  d2 <- d1 - sigma*sqrt(T)
  C <- S0*pnorm(d1)-exp(-r*T)*K*pnorm(d2)
  return(C)
}

#################### ASIAN CALL PRICE - CONTROL VARIATES 1 #####################

AsianCall_CV1 <- function(K,S0,sigma,r,T,delta,n,k){
  # n = number of main simulations
  # k = number of pilot simulations
  m <- T/delta # m = number of time discretization points

  # 1. Pilot Simulation
  C_asian <- rep(0,times=k)
  C_european <- rep(0,times=k)
  for(i in 1:k){
    S <- matrix(0,nrow=k,ncol=m+1)
    S[,1] <- S0
    for(j in 1:m){
      S[i,j+1] <- S[i,j]*exp((r-sigma^2/2)*delta + sigma*sqrt(delta)*rnorm(1))
    }
    C_asian[i] <- exp(-r*T)*max(mean(S[i,-1])-K,0)
    C_european[i] <- exp(-r*T)*max(S[i,m+1]-K,0)
  }

  nu <- BS_Call(K,S0,sigma,r,T) # true price of European Call (control variate)

  var <- sum((C_european - nu)^2)/(k-1)
  cov <- sum((C_asian - mean(C_asian))*(C_european - nu))/(k-1)
  c_hat <- -cov/var
  # sample covariance and variance to obtain optimal value for c

  # 2. Main Simulation
  C_asian <- rep(0,times=n)
  C_european <- rep(0,times=n)

  for(i in 1:n){
    S <- matrix(0,nrow=n,ncol=m+1)
    S[,1] <- S0
    for(j in 1:m){
      S[i,j+1] <- S[i,j]*exp((r-sigma^2/2)*delta + sigma*sqrt(delta)*rnorm(1))
    }
    C_asian[i] <- exp(-r*T)*max(mean(S[i,-1])-K,0)
    C_european[i] <- exp(-r*T)*max(S[i,m+1]-K,0)
  }
  G <- C_asian + c_hat*(C_european - nu)
  G_hat <- mean(G)
  output <- list(G,G_hat)
  return(output)
}

################### EXACT ASIAN GEOMETRIC MEAN CALL PRICE #####################

AsianCall_geom_exact <- function(K,S,sigma,r,T,delta,t){
  # 0=t_0 < t_1 < ... < t_M
  # t_m = m*delta
  # t = time point of price
  # m s.t. t_m <= t <= t_m+1
  # S = (S_t0,...,S_tm)
  M <- T/delta
  time_discr <- seq(0,T,length.out=M+1)
  for(i in 1:(M+1)){
    if(time_discr[i] <= t & t <= time_discr[i+1]){
      m <- i-1
      break
    }
  }

  G <- exp(mean(log(S)))
  nu <- r - sigma^2/2

  a <- m/M * log(G) + (M-m)/M * (log(S[1])+ nu*((m+1)*delta-t) +
                                  1/2*nu*(T-(m+1)*delta))
  b <- (M-m)^2/M^2*sigma^2*((m+1)*delta-t) +
        sigma^2*(T-(m+1)*delta)/(6*M^2)*(M-m)*(2*(M-m)-1)
  x <- (a-log(K)+b)/sqrt(b)

  P <- exp(-r*T)*(exp(a+b/2)*pnorm(x)-K*pnorm(x-sqrt(b)))
  return(P)
}

#################### ASIAN CALL PRICE - CONTROL VARIATES 2 #####################

AsianCall_CV2 <- function(K,S0,sigma,r,T,delta,n,k){
  # n = number of main simulations
  # k = number of pilot simulations
  m <- T/delta # m = number of time discretization points

  # 1. Pilot Simulation
  C_asian_arith <- rep(0,times=k)
  C_asian_geom <- rep(0,times=k)
  for(i in 1:k){
    S <- matrix(0,nrow=k,ncol=m+1)
    S[,1] <- S0
    for(j in 1:m){
      S[i,j+1] <- S[i,j]*exp((r-sigma^2/2)*delta + sigma*sqrt(delta)*rnorm(1))
    }
    C_asian_arith[i] <- exp(-r*T)*max(mean(S[i,-1])-K,0)
    C_asian_geom[i] <- exp(-r*T)*max(exp(mean(log(S[i,-1])))-K,0)
  }

  nu <- AsianCall_geom_exact(K,S0,sigma,r,T,delta,t=0)
  # nu = true price of Geometric Asian Call (control variate)

  var <- sum((C_asian_geom - nu)^2)/(k-1)
  cov <- sum((C_asian_arith - mean(C_asian_arith))*(C_asian_geom - nu))/(k-1)
  c_hat <- -cov/var

  # 2. Main Simulation
  C_asian <- rep(0,times=n)
  C_asian_geom <- rep(0,times=n)

  for(i in 1:n){
    S <- matrix(0,nrow=n,ncol=m+1)
    S[,1] <- S0
    for(j in 1:m){
      S[i,j+1] <- S[i,j]*exp((r-sigma^2/2)*delta + sigma*sqrt(delta)*rnorm(1))
    }
    C_asian_arith[i] <- exp(-r*T)*max(mean(S[i,-1])-K,0)
    C_asian_geom[i] <- exp(-r*T)*max(exp(mean(log(S[i,-1])))-K,0)
  }
  G <- C_asian_arith + c_hat*(C_asian_geom - nu)
  G_hat <- mean(G)
  output <- list(G,G_hat)
  return(output)
}

################################## RESULTS #####################################

# input (slide 135)
seed <- 123
S0 <- 100
r <- 0.01
sigma <- 0.2
T <- 2
delta <- 1/12
K <- 100 #strike
n <- 10000 # number of main simulations
k <- 1000 # number of pilot simulations

# 1. Naive Monte Carlo
set.seed(seed)
result_MC <- AsianCall_MC(K,S0,sigma,r,T,delta,n)
value_MC <- result_MC[[2]]
sd_MC <- sd(result_MC[[1]])/sqrt(n)

#95% confidence interval
alpha <- 0.05
z <- qnorm(1-alpha/2)
lowerbound_MC <- value_MC - z*sd_MC
upperbound_MC <- value_MC + z*sd_MC
width_MC <- 2*z*sd_MC


# 2. Control Variates: European Call
set.seed(seed)
result_CV1 <- AsianCall_CV1(K,S0,sigma,r,T,delta,n,k)
value_CV1 <- result_CV1[[2]]
sd_CV1 <- sd(result_CV1[[1]])/sqrt(n)

#95% confidence interval
alpha <- 0.05
z <- qnorm(1-alpha/2)
lowerbound_CV1 <- value_CV1 - z*sd_CV1
upperbound_CV1 <- value_CV1 + z*sd_CV1
width_CV1 <- 2*z*sd_CV1


# 3. Control Variates: Asian Geometric Call
set.seed(seed)
result_CV2 <- AsianCall_CV2(K,S0,sigma,r,T,delta,n,k)
value_CV2 <- result_CV2[[2]]
sd_CV2 <- sd(result_CV2[[1]])/sqrt(n)

#95% confidence interval
```

```r
alpha <- 0.05
z <- qnorm(1-alpha/2)
lowerbound_CV2 <- value_CV2 - z*sd_CV2
upperbound_CV2 <- value_CV2 + z*sd_CV2
width_CV2 <- 2*z*sd_CV2

################################## COMPARISON #################################
comparison <- matrix(0,nrow=4,ncol=3)
colnames(comparison) <- c("Naive Monte-Carlo", "Control Variates - European",
                          "Control Variates - Asian")
rownames(comparison) <- c("Value","Lower Bound", "Upper Bound", "Width")
comparison[1,] <- c(value_MC, value_CV1, value_CV2)
comparison[2,] <- c(lowerbound_MC, lowerbound_CV1, lowerbound_CV2)
comparison [3,] <- c(upperbound_MC, upperbound_CV1, upperbound_CV2)
comparison[4,] <- c(width_MC, width_CV1, width_CV2)
```

a)

```r
comparison[,1:2]

##             Naive Monte-Carlo Control Variates - European
## Value               7.1680937                   7.1301380
## Lower Bound         6.9470496                   7.0071546
## Upper Bound         7.3891378                   7.2531214
## Width               0.4420881                   0.2459668
```

One can use the standard European call as a control variate where the exact expected value ν can be computed using the Black-Scholes pricing formula. Compared to the naive Monte-Carlo method, using the method of control variates with the European call as a control variate leads to a clear reduction of the variance as the width of the 95% confidence interval is almost halved. However, the width is still pretty large which implies a high variance in the pricing method. This can be traced back to the fact that the control variate does capture the non-linearity of the option payoff, however it lags the path property.

b)

```r
comparison[,2:3]

##             Control Variates - European Control Variates - Asian
## Value                         7.1301380               7.14292453
## Lower Bound                   7.0071546               7.13444373
## Upper Bound                   7.2531214               7.15140532
## Width                         0.2459668               0.01696159
```

Another possibility of a control variate is the geometric-average Asian call (discrete time). Here, the exact price nu can be computed explicitly using the closed-form formula on slide 133. The result shows that compared to the control variate method in part a, the variance is decreased tremendously with a width of the confidence interval of only approx. 0.017 whereas before it was 0.246. Therefore, the geometric-average Asian call is a better choice as control variate than the European call. This can be explained by the fact that it keeps track of non-linearity and path property and thus, gives more useful information about the instrument of interest, the arithmetic-average Asian option.

Exercise 4: in progress

a)

You are given a stock which follows GBM dynamics with S0 = 60, r = 0.01, sigma = 0.4. Suppose we have a down-and-in put option with Maturity in 3 months (i.e., T = 3/12), strike K = 50 and barrier level b = 30. Assume daily monitoring and compute the Monte Carlo price of this option (use daily discretization, i.e. set $\delta t = 1/250$ and simulate price paths by exact simulation).

OTHER VERISON: NAT - other delta

```
  gbm_prices <- function(r,sigma,S0,steps_yearly,T_mat){
    delta <- 1/steps_yearly
    k <- T_mat*steps_yearly #new mat: bc T_mat is time to maturity in years
    S <- 1:(k+1)
    S[1] <- S0
    Z <- rnorm(k)
    for(i in 1:k){
      S[i+1] <- S[i]*exp((r-sigma^2/2)*delta+sigma*sqrt(delta)*Z[i])
    }
    return(S)
  }

  MC_Put_d_i <- function(N,K,b,r,sigma,S0,steps_yearly,T_mat){
    delta <- 1/steps_yearly # discretization
    b_adj <- b*exp(0.5826*sigma*sqrt(delta))  # to increase crossin times

    I_b <- function(x,b){ #indicator for the Put_d_i
      ifelse(any(x<b),1,0)
    }
    Put_d_i <- 1:N
    for(i in 1:N){
      prices <- gbm_prices(r,sigma,S0,steps_yearly,T_mat)
      Put_d_i[i] <- I_b(prices,b_adj)*exp(-r*T_mat)*max(K-prices[length(prices)],0) # payoff
    }
    theta_hat <- mean(Put_d_i)
    crossing_count = sum(Put_d_i!=0)
    return(list(theta_hat=theta_hat,nr_of_crossings=crossing_count))
  }

  set.seed(2000)
  MC_Put_d_i(N=10^4,K=50,b=30,r=0.01,sigma=0.4,S0=60,
  steps_yearly=250,T_mat=3/12)$theta_hat

## [1] 0.01955817
```

```
# barrier indicator function
ind_barr <- function(S,b){
  ifelse(any(S < b),1,0)
}

set.seed(2000)
```

```
# main function
MC_sim_down_in_put <- function(Tmat, r, sigma,S0, h ,b , K, m){

  down_in_put <- function(Tmat, r, sigma,S0, h ,b , K){
    delta <- Tmat/h #this is the difference between the timesteps
    S <- numeric(h) #big question on how to set the timesteps
    S[1]  <- S0
    for( i in 2:h){
        z <- rnorm(1,0,1)
        S[i] <- S[i-1] * exp((r - sigma^2/2)*delta + sigma*sqrt(delta)*z)
    }

    # barrier correction
    b_new <- b*exp(0.5826*sigma*sqrt(delta))
    put <- ind_barr(S,b_new) * exp(-r*Tmat) * max(K - S[length(S)],0)

  }
  MC <- mean(replicate(m,down_in_put(Tmat,r,sigma,S0,h,b,K)))
  return(MC)
}


theta<-MC_sim_down_in_put(Tmat=3/12, r=0.01, sigma=0.4,
                        S0=60, h = 250, b=30 , K=50, m=10000)
theta

## [1] 0.009847178

# comparison with european put
MC_put <- function(S0, K , sigma, Tmat, r, n ){

  sim_put <- function(S0,K, sigma, Tmat, r ){
    z <- rnorm(1,0,1)
    St <- S0 * exp((r - sigma^2/2)*Tmat + sigma*sqrt(Tmat)*z)
    C <- exp(- r *Tmat) * max(K- St,0)
    C
  }

  Call_MC <- mean(replicate(n, sim_put(S0, K , sigma, Tmat, r)))
  return(Call_MC)
}

european_put<- MC_put(S0=60, K=50, sigma=0.4,
                      Tmat=3/12, r= 0.01, n=10000)
european_put

## [1] 1.052294
```

Daily monitoring was used assuming that each year has 250 trading days. we are considering a quarter of a year for our simulation. 10000 MC simulations where performed and a correction for the barrier as in the slides was applied since we are in a situation of discrete monitoring and not correcting would mean a lower chance to cross the barrier. It is evident that the price of the down and in option is much lower than the price for a corresponding European Option Ceteris Paribus. This is due to the fact that given a Strike of 60 and a barrier of 30, a barrier crossing will be a rare event and hence the chance of gaining a payoff is lower. This implies a cheap price for the down and in option.

b)

Is barrier crossing a rare event here? Explain. If that is the case, use importance sampling to improve the Monte Carlo price estimates for the barrier option. If not, explain in which situations and how one can use importance sampling to improve the Monte Carlo price estimates for a barrier option.

A barrier crossing event can be considered as a rare event since we are working with a down-and-in-option and the initial value $S_0 = 60$ is considerably higher than the barrier or 30. Adding to this, the volatility is low, which leads to lower chances of crossing the barrier. This is why most replications will give a payoff of 0 which will affect the estimate of the real price in the simulation. Hence, our Monte Carlo estimates can benefit from importance sampling through an exponential change of measure, which can increase the chance of a knock-in.

Indeed we look at our result from part a, we see that we get

```
MC_Put_d_i(N=10^4,K=50,b=30,r=0.01,sigma=0.4,S0=60,
steps_yearly=250,T_mat=3/12)$nr_of_crossings
```

```
## [1] 8
```

that the boundary was only crossed 8 times in 10000 simulations.
The idea is to change the drift such that we drive the GBM distribution to regions where it is more likely to cross the barrier. We start by considering our underlying asset modelled as a GBM, where the price is dependent on the full path of $S = (S_1, \ldots, S_m)$ and simulated in a discrete way via:

$$S_j = S_{j-1} exp(r - \sigma^2/2)h + \sigma\sqrt{h}Z_j$$

where $h = t_j - t_{j-1}$ are the equidistant time steps and $Z_j \sim N(0, 1)$.

In the multivariate case, instead of of drawing from the distribution $Z_j \sim N(0, I_n)$ we can draw from $Z_j \sim N(\mu, I_n)$ and compute the price under the new distribution with shifted mean μ as:

$$E(h(Z)) = E_g(\frac{h(Z)f(Z)}{g(Z)})$$

with f and g being resp. the distributions of $N(0, I_n)$ and $N(\mu, I_n)$. In particular:

$$g(Z) = f(x)e^{-\mu' Z + 1/2\mu'\mu)}$$

We then plug in the two distributions in the previous formula to obtain:

$$E(h(Z)) = E_g(\frac{h(Z)e^{-1/2ZI'Z}}{e^{-1/2(Z-\mu)I'(Z-\mu)}}) = E_g(h(Z)e^{-\mu' Z + 1/2\mu'\mu)})$$

Hence Z is now drawn from a different distribution which will now be shifted toward a mean level that corresponds to the region when the chances of barrier crossing are higher. To complete the equation above, in the case of a down and in put option, the payoff can be expressed as a function of Z and is :

$$h(Z) = h(Z_1, \ldots, Z_m) = e^{-rT}(I(S))(K - S_m)^+$$

where I(S) is 1 when $S_j < b$ and 0 otherwise. Hence our drift-shifting equation becomes:

$$E(h(X)) = E_g(e^{-rT}(I(S))(K - S_m)^+ e^{-\mu' X + 1/2\mu'\mu)})$$

We may now simulate as follows:

```
1. for replications $i = 1 : N$
    (a) generate $Z_i \sim N(\mu, I)$
    (b) $Y_i < -G(Z(i))exp(\mu' Z_i + 1/2\mu'\mu)$
2. $\theta < -1/n \sum_{i=1}^{N} Y_i$
```

```
# CODE?
```