Question 1

Part 1)

Monte Carlo

The results obtained are shown below with N = 300 and M = 100000

| | Price | Standard Error | Standard Deviation | Time (Seconds) |
|---|---|---|---|---|
| Call | 9.093092 | 0.0432043 | 13.6624 | 138.10 |
| Put | 6.296585 | 0.02877055 | 9.098046 | 140.53 |

Part 2)

European Call options

| | MC | Antithetic Variates (N = 300, M = 50000) | Delta Based Controlled Variates (N = 300, M = 10000) | Antithetic Variates and Delta Based Controlled Variates |
|---|---|---|---|---|
| Call Option | 9.093092 | 1.069085 | 9.137195 | 9.143035 |
| Standard Error | 0.0432043 | 0.0029 | 0.005868151 | 0.0502 |
| Standard Deviation | 13.6624 | 0.9294 | 0.5868 | 5.026795 |
| Time (minutes) | 2.30 | 7.78 | 10.49 | 43.80 |

European Put Options

| | MC | Antithetic Variates | Delta Based Controlled Variates (N = 300, M = 10000) | Antithetic Variates and Delta Baes Controlled Variates |
|---|---|---|---|---|
| Put Option | 6.296585 | 0.1174082 | 6.343732 | 6.3039 |
| Standard Error | 0.02877055 | 0.000481 | 0.19273 | 0.0271189 |
| Standard Deviation | 9.098046 | 0.1524194 | 19.2732 | 2.71189 |
| Time (minutes) | 2.34 | 5.035 | 17.19 | 40.17 |

The tables above show the results obtained for European Call and Put options. As seen all the methods except for Antithetic produce the similar price for the call and put option. The taken however varies a lot with MC being the lowest and the mixture of Antithetic Variates and Delta Based Controlled Variates being the highest.

Question 2

The table below shows the results obtained using different methods for the number of steps and simulations shown below. As seen, the prices, bias, RMSE and the time taken is consistent among the different methods.

N = 500, M = 10000

| Method | Price | Bias | RMSE | Time(minutes) |
|---|---|---|---|---|
| Absorption | 6.84289 | 0.2589045 | 7.80192 | 2.97 |
| Reflection | 6.824502 | 0.2396127 | 7.81462 | 3.27 |
| Higham and Mao | 6.928372 | 0.34685 | 7.75639 | 3.98 |
| Partial truncation | 6.874858 | 0.2916014 | 7.673852 | 3.16 |
| Full truncation | 6.488217 | 0.2640965 | 7.655728 | 3.13 |

Question 3

Part a)

The Cholesky decomposition of the matrix A is

```
1  0.5000000  0.2000000
0  0.8660254 -0.5773503
0  0.0000000  0.7916228
```

Part C)

The stock was simulated in a 3-dimensional matrix m times. The price obtained for the European Call and Put basket option are shown below

European Call: 2.058736

European Put: 1.497829

Part d)

The exotic option was priced with a barrier of 104. The price obtained is 2.07

Appendix (R code)

```
#Question 1  -----
# Part 1
BSMC <- function(S0,k,r,tau,sigma,div,N,M, isCall){
    start_time = Sys.time()
    cp = ifelse(isCall,1,-1 )
    dt = tau / N
    nudt = (r-div-0.5*sigma^2)*dt
    sig = sigma * sqrt(dt)
    Sum1 = 0
    Sum2 = 0
    S = matrix(0, nrow = M, ncol = (N+1))
    w = matrix(0, nrow = M, ncol = N)
    S[,1] = log(S0)
    j = 1
    i =2
    for(j in 1:M){
     w[j,] = rnorm(1,mean = 0, sd = sqrt(dt))
    for(i in 2:(N+1)){
            S[j,i] = S[j,i-1] + nudt + sig*w[j,i-1]
    }
    }
     # change these to matrices
     ST = c()
     CT = c()
     ct = c()
     i = 1
     for(i in 1:M){
```

```
      ST[i] = exp(S[i,N+1])
      CT[i] = max(0,cp*(ST[i] - k))
      ct[i] = CT[i]^2
    }

      Sum1 =  sum(CT)
      Sum2 =  sum(ct)

   option_value = Sum1/M * exp(-r*tau)
   SD = sqrt((Sum2 - Sum1^2/M)*exp(-2*r*tau)/(M-1))
   SE = SD/sqrt(M)
   end_time = Sys.time()
   Time = end_time - start_time
   list(Call_Price = option_value, Standard_Deviation = SD, Standard_Error = SE, Total_Time = Time)
}
S0 = k = 100
r = 0.06
tau = 1
sigma = 0.2
div = 0.03
N = 300
M = 100000
BSMC( S0,k,r,tau,sigma,div,N,M,T)
BSMC(S0,k,r,tau,sigma,div,N,M,F)



# Antithetic
ABSMC<- function(S0,k,r,tau,sigma,div,N,M, isCall){ # Anthithetic variates
```

```
start_time = Sys.time()
cp = ifelse(isCall,1,-1 )



dt = tau / N
nudt = (r-div-0.5*sigma^2)*dt
sig = sigma * sqrt(dt)
Sum1 = 0
Sum2 = 0

S1 = S2 = matrix(0, nrow = M, ncol = (N+1))
w = matrix(0, nrow = M, ncol = (N))

S1[,1] = log(S0)
S2[,1] = log(S0)
i = 2
j = 1
for(j in 1:M){
w[j,] = rnorm(N,mean = 0, sd = sqrt(dt))
    for(i in 2:(N+1)){
      S1[j,i] = S1[j,(i-1)] + nudt + sig*w[j,(i-1)]
      S2[j,i] = S2[j,(i-1)] + nudt + sig*(-w[j,(i-1)])
   }
}
   CT = ST1 = ST2 = ct = c()
   i = 1
   for(i in 1:M){
   ST1[i] = exp(S1[i,(N+1)])
```

```
        ST2[i] = exp(S2[i,(N+1)])
        CT[i] = 0.5*(max(0,cp*(ST1[i] - k)) + max(0,cp*(ST2[i] - k)))
        ct[i] = CT[i]^2
        }
        Sum1 = sum(CT)
        Sum2 = sum(ct^2)
    option_value = Sum1/M * exp(-r*tau)
    SD = sqrt((Sum2 - Sum1^2/M)*exp(-2*r*tau)/(M-1))
    SE = SD/sqrt(M)
    end_time = Sys.time()
    Time = end_time - start_time
    list(Price = option_value, Standard_Deviation = SD, Standard_Error = SE, Time = Time)
}
N = 300
M = 100000

ABSMC(S0,k,r,tau,sigma,div,N,M,T)
ABSMC(S0,k,r,tau,sigma,div,N,M,F)




# Delta function
Delta <- function(S,K,t,r,sig,div,Tm,isCall){

    tau<-Tm-t
    d1<-(log(S/K)+((r-div+((sig*sig)/2))*tau))/(sqrt(tau)*sig)

    if(isCall)
```

```
        return(exp(-div*tau)*pnorm(d1))
    else
        return(exp(-q*tau)*(pnorm(d1)-1))
}

# Delta based controlled variate

DeltaBased <- function(S0,k,r,tau,sigma,div,N,M, isCall){
    start_time = Sys.time()
    cp = ifelse(isCall,1,-1 )

    dt = tau / N
    nudt = (r-div-0.5*sigma^2)*dt
    sig = sigma * sqrt(dt)
    erdt = exp((r-div)*dt)
    beta = -1

    Sum1 = 0
    Sum2 = 0
    St = cv = CV =ST= c()
    St[1] = S0
    cv[1] = 0
    CT = c()
    w = c()

    for(j in 1:M){
        for(i in 2:(N+1)){
            t = (i - 1) *dt
```

```
        delta = Delta(St[i-1],k,t,r,sigma,div,tau,cp)
        w[i] = rnorm(N, mean = 0, sd = 1)
        Stn = St[i-1] *exp(nudt + sig*w[i])
        cv[i] = cv[i-1] + delta *(Stn - St[i-1] * erdt)
        St[i] = Stn
      }
      ST[j] = St[(N+1)]
      CV[j] = cv[(N+1)]
      CT[j] = max(0 , cp*(ST[j] - k)) + beta * CV[j]

    }
    Sum1 = sum(CT)
    Sum2 = sum(CT^2)

    option_value = Sum1/M * exp(-r*tau)
    SD = sqrt((Sum2 - Sum1^2/M)*exp(-2*r*tau)/(M-1))
    SE = SD/sqrt(M)

    end_time = Sys.time()
    Time = end_time - start_time
    list(Call_Price = option_value, Standard_Deviation = SD, Standard_Error = SE, Time = Time)
}

S0 = k = 100
r = 0.06
tau = 1
sigma = 0.2
div = 0.03
```

```
DeltaBased(S0,k,r,tau,sigma,div,N,M,T)
DeltaBased(S0,k,r,tau,sigma,div,N,M,F)



DeltaAnti <- function(S0,k,r,tau,sigma,div,N,M, isCall){
  start_time = Sys.time()

  dt=tau/N
  nudt=(r-div-0.5*sigma^2)*dt
  sig<-sigma*sqrt(dt)
  erddt<-exp((r-div)*dt)
  cp = ifelse(isCall,1,-1 )
  beta1=-1

  Sum1=0
  Sum2=0

  ST1= ST2 = cv1 = cv2 = st1 = st2 = CV1 = CV2  = w = CT = c()
  ST1[1] = ST2[1] = S0
  cv1[1] = cv2[1] =0

  for(j in 1: M){
  for(i in 2:(N+1)){
    t=(i-1)*dt

    delta1<-Delta(ST1,k,t,r,sig,div,tau,isCall)
    delta2<-Delta(ST2,k,t,r,sig,div,tau,isCall)
```

```
    w[i] <-rnorm(N,mean =0, sd = 1)


    Stn1<-ST1[i-1]*exp(nudt+sig*w[i])


    Stn2<-ST2[i-1]*exp(nudt+sig*(-w[i]))


    cv1[i] <-cv1[i-1] + delta1 * (Stn1-ST1[i-1] *erddt)
    cv2[i] <-cv2[i-1] + delta2 * (Stn2-ST2[i-1]*erddt)


    ST1[i] = Stn1
    ST2[i] = Stn2
}
st1[j] = ST1[(N+1)]
st2[j] = ST2[(N+1)]
CV1[j] = cv1[(N)]
CV2[j] = cv2[(N)]
if(isCall == "F"){
CT[j]=0.5*(max(0,(k - st1[j]))+(beta1*CV1[j]) + max(0,(k - st2[j]))+(beta1*CV2[j]))
} else if(isCall == "T"){
    CT[j]=0.5*(max(0,(st1[j]- k))+(beta1*CV1[j]) + max(0,(st2[j]- k))+(beta1*CV2[j]))


}
}
Sum1 = sum(CT)
Sum2 = sum(CT^2)

option_value = Sum1/M * exp(-r*tau)
SD = sqrt((Sum2 - Sum1^2/M)*exp(-2*r*tau)/(M-1))
```

```
  SE = SD/sqrt(M)

  end_time = Sys.time()

  Time = end_time - start_time

  list(Option_Price = option_value, Squared_Error = SE, Standard_Deviation = SD, Time = Time)

}

S0 = k = 100

r = 0.06

tau = 1

sigma = 0.2

div = 0.03

N = 300

M = 10000


DeltaAnti(S0,k,r,tau,sigma,div,N,M,T)





#Question 2 ----


Heston <- function(N,M,type){

  # functions f1

  start_time = Sys.time()

  f1 <- function(x,type){


    if(type == "A"){

      ans = max(0,x)

      return(ans)
```

```
    }
    if(type == "R"){
        ans = abs(x)
        return(ans)
    }
    else{
        return(x)
    }
}
f2 <- function(x,type){
    if(type == "A" || type =="F"){

        ans = max(0,x)
        return(ans)
    }
    if(type == "R"){
        ans = abs(x)
        return(ans)
    }
    else{
        return(x)
    }
}
f3 <- function(x,type){
    if(type == "A"|| type =="F" || type =="P"){
        ans = max(0,x)
        return(ans)
    }
```

```
    if(type == "R" || type =="H"){
        ans = abs(x)
        return(ans)
    }
}



# initial parameters

S0 = 100
k = 100
kappa = 6.21
theta = 0.019
V0 = 0.010201
alpha = 0.5
n = N
tm = 1
dt = tm/n
lambda = 1
beta = 1
rho = -0.7
sigma = 0.61
r=0.0319
Value = 6.8061

Lnst =  c()
V = V_tilda = c()
V_tilda[1] = V[1] = V0
```

```
wv = ws = c()

for(j in 1:M) {
  z = matrix(0,nrow = 2, ncol = (N+1))
  z[1,] = rnorm((N+1))
  z[2,] = rnorm((N+1))

  cor = matrix(data = c(1,rho,rho,1),2,2)  # correlation matrix
  cor_chol = t(chol(cor))
  dat = cor_chol %*% z
  ws = dat[1,]
  wv = dat[2,]



  for(i in 2:(N+1)){
    V_tilda[i] = f1(V_tilda[i-1],type) - kappa * dt *(f2(V_tilda[i-1],type) - theta)+ sigma *
      (f3(V_tilda[i-1],type))^alpha *(wv[i-1]) * sqrt(dt)
    V[i]= f3(V_tilda[i],type)
  }

  lnst = c()
  lnst[1] = log(S0)

  for(i in 2:(N+1)){
    lnst[i] = lnst[(i-1)] + (r-0.5*V[(i-1)])* dt +
      sqrt(V[(i-1)]*dt) * ws[(i-1)]
  }
```

```
        Lnst[j] = lnst[N+1]
    }
    St = Ct = Ct2 =  c()

    for(i in 1:M){
        St[i] <- exp(Lnst[i])
        Ct[i] <- max(0,(St[i]-k))
        Ct2[i] = Ct[i] ^2
    }
    Sum1 <- sum(Ct)
    Sum2 = sum(Ct2)

    Price <- (Sum1/M)*exp(-r*tm)

    bias = mean(Ct) - Value
    RMSE = sqrt(bias^2 + var(Ct))
    end_time = Sys.time()
    Time = end_time - start_time
    list(Calculated_Price = Price, Root_Mean_Squared_Error = RMSE, Bias = bias, Time = Time)


}
# Types : A,R,F,P,H

N = 500
M = 10000
```

```
Ref = Heston(N,M,"R")
Ab = Heston(N,M,"A")
Ft = Heston(N,M,"F")
Pt= Heston(N,M,"P")
Hig = Heston(N,M,"H")




# Question 3 ----
A = matrix(c(1.0,0.5,0.2,0.5,1.0,-0.4,0.2,-0.4,1.0),3,3)
cholA = chol(A)
cholA
t_CholA = t(cholA)


nu = c(0.03,0.06,0.02)
sig = c(0.05,0.2,0.15)
S0 = c(100,101,98)
tau = 100/365
dt = 1/365
N = tau/dt
M = 1000

S = array(0,dim = c(M,N,3))
S[,1,1] = log(S0[1])
S[,1,2] = log(S0[2])
```

```
S[,1,3] = log(S0[3])
W = matrix(0,nrow = N, ncol = 3)



for(l in 1:3){
  for(j in 1:M){
    W[,1] = rnorm(N)
    W[,2] = rnorm(N)
    W[,3] = rnorm(N)
    Z =t_CholA %*% t(W)
    Ze <- t(Z)
    #colnames(Z) <- c("W1","W2","W3")
    for(i in 2:(N)){
      S[j,i,l] = S[j,i-1,l] + (nu[l] - (0.5 * sig[l]^2))* dt + (sig[l] * Ze[i,l] * sqrt(dt))
    }
  }
}
a1 = a2 = a3 = 1/3

S1 = exp(S[,100,1])
S2 = exp(S[,100,2])
S3 = exp(S[,100,3])

# Plot
plot3d(S[,1,1],S[,1,2],S[,1,3])

U = c()
```

```
for(i in 1:M){
   U[i] = a1*S1[i] + a2*S2[i] + a3*S3[i]
}
avg_U = mean(U)

Call = c()
Put = c()
k = 100
for(i in 1:length(U)){
   Call[i] = max((U[i] - k),0)
   Put[i] = max((k - U[i]),0)
}
mean(Call)
mean(Put)
r = mean(nu)
Call_Value = mean(Call)*exp(-r*tau)
Put_Value = mean(Put)*exp(-r*tau)

B_payoff = 0
B = 104
bool = S[,,2] >= B
"TRUE" %in% bool

for(j in 1:M){
   if( B <= S[j,,2]){
      B_payoff = max(U[i]-k,0)


   }
```

```
    else if(max(S[j,,2]) > max(S[j,,3])){
        B_payoff = max(0,(S[j,,2] - k)^2)

    }
    else if(mean(S[j,,2]) > mean(max(S[j,,3]))){
        B_payoff = mean(0,(S[j,,2] - k))

    }
    else {
        B_payoff =  mean(Call)*exp(-r*tau)

    }
}
Price = B_payoff* exp(-r *tau)
Price
```