FE621 - Final

Napat L

May 9, 2018

Problem A: Asian Option Pricing using Monte Carlo Control Variate.

(A)

The analytic solution of geometric Asian option in the Black-Scholes model is 15.17119.

```
geoAsianOpt(S0=100, sigma=0.3, K=100, r=0.03, Tm=5, Nt=5*252, type=0)

## [1] 15.17113
```

(B)

The Monte Carlo price of an arithmetic Asian call option with 1,000,000 simulations is 17.43436.

```
MonteCarloAriAsian(TRUE, S0=100, K=100, T=5, sig=0.3, div=0, r=0.03, Nt = 5*252, M = 1000000)

## $opt_value
## [1] 17.43436
##
## $SE
## [1] 0.03069516
##
## $time
## elapsed
##  310.47
```

(C)

The Monte Carlo price of a geometic Asian call option with 1,000,000 simulations is 15.17123 with 0.02162398 standard error.

```
MonteCarloGeoAsian(TRUE, S0=100, K=100, T=5, sig=0.3, div=0, r=0.03, Nt = 5*252, M = 1000000)

## $opt_value
## [1] 15.17123
##
## $SE
## [1] 0.02162398
##
## $time
## elapsed
##  350.72
```

(D)

Using 10,000 simulations, we obtain beta between arithmetic and geometric Asian Option equal to 1.153287 with 0.03087273.

```
data.frame(MCCVAsianOpt(S0=100, sigma=0.3, K=100, r=0.03, Tm=5, Nt=5*252, M1=100000, M2=10000, type=0))

##   geo.Price ari.Price time1        b CV.price   se.error time2 total.time
## 1  15.17094  17.49831 136.1 1.153287 17.48776 0.03087273  0.81     136.91
```

(E), (F) and (D)

With arithmetic Asian option control variate by geometric Asian based, we obtain the results using 3 different simulations as below.

- Using 100,000 simulations, we obtain control variate price equal to 17.4715 with very small 0.00954 standard error.

```
data.frame(MCCVAsianOpt(S0=100, sigma=0.3, K=100, r=0.03, Tm=5, Nt=5*252, M1=100000, M2=100000, type=0))

##   geo.Price ari.Price time1        b CV.price    se.error time2 total.time
## 1  15.16493  17.46437   223 1.149384  17.4715 0.009539912  7.18     230.18
```

- Using 10,000 simulations, we obtain control variate price equal to 17.45122 with very small 0.02842129 standard error.

```
data.frame(MCCVAsianOpt(S0=100, sigma=0.3, K=100, r=0.03, Tm=5, Nt=5*252, M1=10000, M2=10000, type=0))

##   geo.Price ari.Price time1        b CV.price   se.error time2 total.time
## 1  14.77277  16.99471   3.7 1.145992 17.45122 0.02842129  0.64       4.34
```

- Using 1,000 simulations, we obtain control variate price equal to 17.47611 with very small 0.0962493 standard error. We can conclude that using control variate method makes the price stable.

```
data.frame(MCCVAsianOpt(S0=100, sigma=0.3, K=100, r=0.03, Tm=5, Nt=5*252, M1=1000, M2=1000, type=0))

##   geo.Price ari.Price time1        b CV.price   se.error time2 total.time
## 1  15.97615  18.38798  0.65 1.132732 17.47611 0.0962493  0.07       0.72
```

(BONUS)

Using Bloomberge terminal Asian option data for IBM with 1.7% interest rate and 0% dividend, we have Bloomberge's prices as below:

Y-Axis	X-Axis	X-Axis	X-Axis	X-Axis	X-Axis	X-Axis
Strike	Maturity	12M	24M	36M	48M	60M
144.15	Price (Total)	6.71	10.32	13.15	15.51	17.58
	Volatility	19.907%	21.164%	21.356%	21.470%	21.701%
149.15	Price (Total)	4.42	7.94	10.76	13.14	15.23
	Volatility	19.557%	20.936%	21.141%	21.258%	21.484%
154.15	Price (Total)	2.78	5.99	8.71	11.04	13.12
	Volatility	19.197%	20.681%	20.913%	21.065%	21.313%
159.15	Price (Total)	1.69	4.44	6.98	9.22	11.25
	Volatility	18.979%	20.500%	20.696%	20.847%	21.084%
164.15	Price (Total)	1.03	3.27	5.56	7.68	9.62
	Volatility	18.770%	20.316%	20.486%	20.681%	20.901%
169.15	Price (Total)	0.63	2.39	4.42	6.38	8.21
	Volatility	18.676%	20.168%	20.349%	20.520%	20.778%

When using our functions to price the same parameters as we do with Bloomberge terminal, we obtain the list of prices as below:

```
##                1Y        2Y        3Y        4Y        5Y
## K:144.15 7.1451626 10.932994 13.685807 16.035260 18.192618
## K:149.15 4.8544920  8.613519 11.358936 13.707806 15.885865
## K:154.15 3.1204799  6.645391  9.315582 11.642767 13.822504
## K:159.15 1.9258085  5.059324  7.551500  9.806392 11.930402
## K:164.15 1.1284960  3.781289  6.057151  8.208802 10.257994
## K:169.15 0.6433711  2.780717  4.828737  6.834842  8.807438
```

As we can see, the prices are similar but they are not that close to each other.

Problem B: Parameter estimate for Stochastic Differential Equation.

(1)

Using AIC to select the best model for all of 5 stocks, we have that the first model is the best fit for stock 1, 2, 3 and 5 but the fifth model is the best fit for stock 4.

```
best.model

##              best.model
## stock1   model1.euler
## stock2   model1.euler
## stock3   model1.euler
## stock4   model5.euler
## stock5   model1.kessler
```

(2)

Using AIC, BIC and Log likelihood on first model of stock 1, they define the same result that the best method is Euler.

```
table1 <- EstimateParameter(data[,1], 1)
table1

##    method      theta1     theta2     theta3       AIC       BIC   LogLik
## 1   euler 0.008062477 0.04298704  0.5972266 -256393.2 -129581.3 128199.6
## 2   ozaki 0.008076305 0.04348005  0.5952456 -256391.5 -129579.3 128198.8
## 3   shoji 0.008076321 0.04348001  0.5952458 -256391.5 -129579.1 128198.8
## 4 kessler 0.635704087 0.45241821 -1.2589008       6.0  23.02587      0.0
```

Using AIC, BIC and Log likelihood on first model of stock 2, they define the same result that the best method is Euler.

```
table2 <- EstimateParameter(data[,2], 1)
table2

##    method      theta1     theta2    theta3       AIC       BIC   LogLik
## 1   euler 0.006692336 0.03178109 0.7903063 -129598.3 -129581.3 64802.15
## 2   ozaki 0.006746619 0.03198329 0.7891716 -129596.4 -129579.3 64801.18
## 3   shoji 0.006652748 0.03200848 0.7890695 -129596.2 -129579.1 64801.08
## 4 kessler 0.007186735 0.03107624 0.7945091 -129595.8 -129578.8 64800.90
```

Using AIC, BIC and Log likelihood on first model of stock 3, they define the same result that the best methods are Euler and Kessler. However, since AIC, BIC and Log likelihood make very bad result, this means the first model is not fit enough for stock 3. We should find other type of model to re-evaluate.

```
table3 <- EstimateParameter(data[,3], 1)
table3

##    method      theta1      theta2    theta3      AIC         BIC    LogLik
## 1   euler-6.447687616 -1.24002471 -1.037941     6.00    23.02587      0.00
## 2   ozaki 0.007479773 -0.01127238  1.091292 30019.04 30036.06919 -15006.52
```

```
## 3   shoji 0.007676594 -0.01128092  1.091050 30020.89 30037.91681 -15007.45
## 4 kessler 0.690893265  0.48781465 -1.131420     6.00    23.02587      0.00
```

Using AIC, BIC and Log likelihood on first model of stock 4, they define the same result that the best method is Euler.

```
table4 <- EstimateParameter(data[,4], 5)
table4

##    method      theta1     theta2    theta3     theta4       AIC       BIC   LogLik
## 1   euler 0.003274155 -0.3106005 0.08065590 0.6963570 -130496.7 -130481.7 65252.36
## 2   ozaki 0.003551525 -0.3628812 0.08852893 0.6816740 -130496.4 -130481.4 65252.22
## 3   shoji 0.003499591 -0.3642739 0.08862773 0.6816288 -130496.4 -130481.4 65252.22
## 4 kessler 0.003620323 -0.3392665 0.08523160 0.6874293 -130496.5 -130481.5 65252.24
```

Using AIC, BIC and Log likelihood on first model of stock 5, they define the same result that the best method is Kessler.

```
table5 <- EstimateParameter(data[,5], 1)
table5

##    method      theta1     theta2    theta3       AIC       BIC   LogLik
## 1   euler 0.006347047 0.04494081 0.7893863 -54430.38 -54413.36 27218.19
## 2   ozaki 0.006271558 0.04507093 0.7886705 -54429.62 -54412.59 27217.81
## 3   shoji 0.006231659 0.04502249 0.7888340 -54429.69 -54412.67 27217.85
## 4 kessler 0.006272561 0.04430150 0.7919271 -54432.37 -54415.34 27219.18
```

(3)

From the previous question, we summarize the result as below.

Stock	Method
1	Euler
2	Euler
3	Euler/Kessler
4	Euler
5	Kessler

As we can see, Euler makes the best model estimation for 4 times, but Kessler makes 2 times. Therefore, Euler is the best estimate method in this case.

Problem C: Principal Component Analysis.

(1)

We download 30 components in Down Jones and calculate the standardized returns. We obtain the result as below:

```
return.std <- apply(ReturnMatrix, 2, FUN = function(x){(x - mean(x))/sd(x)})
head(return.std)[,1:5]
```

```
##                    MMM         AXP       AAPL         BA         CAT
## 2013-05-09  1.56569543 -0.07362986 -0.5194170  0.3299524 -0.05909781
## 2013-05-10  0.43865421 -0.17376743 -0.6368585 -0.3596082 -1.03761708
## 2013-05-13 -0.03680290 -0.35235551  0.2039650  0.3140518 -0.45672451
## 2013-05-14  0.02449046  1.89569894 -1.7100993  0.9376944 -0.43628082
## 2013-05-15  0.74190436  1.35791362 -2.3903046  0.5963000 -0.46186008
## 2013-05-16 -0.42748117 -0.61937299  0.8540858 -0.4042112 -0.20773120
```

(2)

We use standardized returns to calculate the covariance matrix. As we can see, the variance of returns after standardization, they become equal to 1.

```
cov.return <- cov(return.std)
head(cov.return)[,1:7]
```

```
##            MMM       AXP      AAPL        BA       CAT       CVX      CSCO
## MMM  1.0000000 0.4412873 0.3521350 0.5084108 0.4982883 0.4330846 0.4688317
## AXP  0.4412873 1.0000000 0.2525758 0.4023464 0.4106654 0.3403041 0.3377412
## AAPL 0.3521350 0.2525758 1.0000000 0.3285351 0.3420651 0.2643806 0.3600394
## BA   0.5084108 0.4023464 0.3285351 1.0000000 0.4411875 0.3713001 0.3686770
## CAT  0.4982883 0.4106654 0.3420651 0.4411875 1.0000000 0.5246057 0.3938438
## CVX  0.4330846 0.3403041 0.2643806 0.3713001 0.5246057 1.0000000 0.3749529
```

(3)

We calculate and show top 5 eigenvalues. As we can see, when we summarize the biggest 5 eigenvalues, we have that they are 56.368% of all 30 eigenvalues.

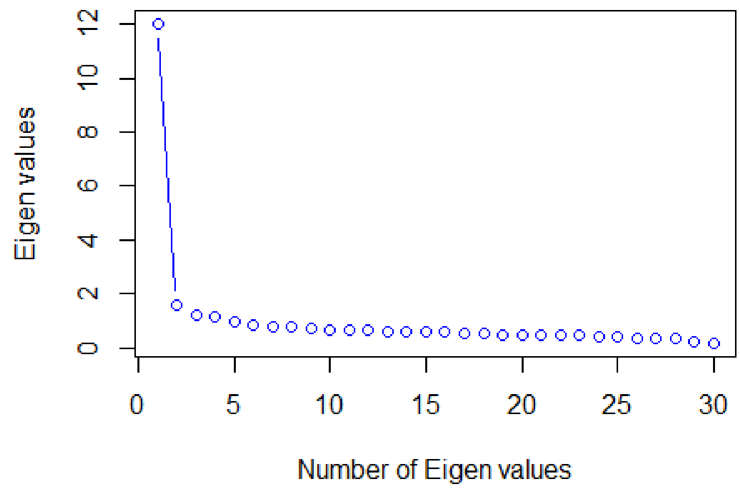
```
eigen.return <- eigen(cov.return)
head(eigen.return$values)

## [1] 12.0243109  1.5632486  1.2334006  1.1343554  0.9549452  0.8265063

print(sum(eigen.return$values[1:5])/sum(eigen.return$values))*100

## [1] 56.36754

plot(1:length(eigen.return$values),eigen.return$values, type = "b", col = "blue", ylab = "Eige
n values", xlab = "Number of Eigen values")
```



(4)

After we calculate Ft, we compute mean and sd of Ft. We obtain mean equal to -0.05621707 and sd equal to 1.

```
mean.F1

## [1] -0.05621707

sd.F1

## [1] 1
```

(5)

After linear regression between DIA and Ft, we obtain the R-squared equals to 0.975. It means that Ft and DIA have very high linear relationship.

```
getSymbols("DIA", from="2013-05-09", src="yahoo")

## [1] "DIA"

return.dia <- dailyReturn(DIA)

return.dia.std <- apply(return.dia, 2, FUN = function(x){(x - mean(x))/sd(x)})

lm.dia1 <- lm(return.dia.std ~ F1)
summary(lm.dia1)

##
## Call:
## lm(formula = return.dia.std ~ F1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.63078 -0.08492  0.00506  0.09476  0.65670
```

```
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.055510   0.004459  -12.45   <2e-16 ***
## F1          -0.987423   0.004454 -221.70   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1582 on 1260 degrees of freedom
## Multiple R-squared:  0.975,  Adjusted R-squared:  0.975
## F-statistic: 4.915e+04 on 1 and 1260 DF,  p-value: < 2.2e-16
```

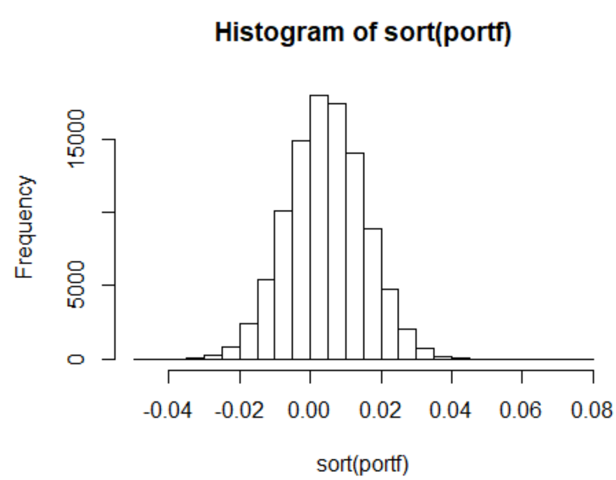
(6)

We calculate beta of factors for 30 stocks. We obtain and show the sample as below:

```
head(beta30)

##          betaF1       betaF2      betaF3       betaF4       betaF5
## MMM  -0.7512603  0.004066882 -0.03243400 -0.036340835 -0.030478791
## AXP  -0.6294010 -0.170912799  0.22685411  0.251236583 -0.072122846
## AAPL -0.4875150 -0.108161688  0.23106076 -0.409769574  0.164485174
## BA   -0.6445013 -0.142000435  0.07497703 -0.004192582 -0.185163378
## CAT  -0.6370758 -0.384798942 -0.17089131 -0.107822273 -0.065531641
## CVX  -0.6435266 -0.247333454 -0.52211761 -0.031079515 -0.006787084
```

Using factor model for stock return with 100,000 simulations, we obtain the distribution of return as below:



We calculate VaR and obtain that the daily VaR equals to 2.06142% and weekly VaR equal to 5.61375%.

```
VaR.daily <- quantile(sort(portf), probs = 0.01)*100
VaR.daily

##        1%
## -2.061417
```

```r
Var.weekly <- quantile(sort(portf), probs = 0.01)*100*sum(sigma30.bar/30)*100*sqrt(5)
Var.weekly

##        1%
## -5.613745
```

Appendix

Problem A – A

```
geoAsianOpt <- function(S0, sigma, K, r, Tm, Nt, type){

    adj_sigma <- sigma*sqrt((2*Nt+1)/(6*(Nt+1)))
    rho <- 0.5*(r-(sigma^2)*0.5+adj_sigma^2)

    d1 <- (log(S0/K)+(rho+0.5*adj_sigma^2)*Tm)/(adj_sigma*sqrt(Tm))
    d2 <- (log(S0/K)+(rho-0.5*adj_sigma^2)*Tm)/(adj_sigma*sqrt(Tm))

    if(type == 0){
        price <- exp(-r*Tm)*(S0*exp(rho*Tm)*pnorm(d1)-K*pnorm(d2))
    }else{
        price <- exp(-r*Tm)*(-S0*exp(rho*Tm)*pnorm(-d1)+K*pnorm(-d2))
    }
    return(price)
}
```

Problem A – B

```
MonteCarloAriAsian <- function(isCall, S0, K, T, sig,div,r,Nt,M){
    dt <- T/Nt
    nudt <- (r-div-0.5*sig^2)*dt
    sigsdt <- sig*sqrt(dt)
    lns <- log(S0)
    sum_OT <- 0
    sum_OT2 <- 0
    start.time <- proc.time()

    for (j in 1:M){
        w <- rnorm(Nt)
        lnSt <- lns
        sum_ST = 0
        for(i in 1:Nt){
            lnSt <- lnSt + nudt + sigsdt*w[i]
            St <- exp(lnSt)
            sum_ST = sum_ST + St
            }
        OT <- ifelse(isCall, max(0,sum_ST/(Nt+1)-K), max(0,K-sum_ST/(Nt+1)))
        sum_OT <- sum_OT+OT
        sum_OT2 <- sum_OT2+OT*OT
        }
    opt_value <- sum_OT/M*exp(-r*T)
    SD <- sqrt((sum_OT2 - sum_OT*sum_OT/M)*exp(-2*r*T)/(M-1))
    SE <- SD/sqrt(M)
    end.time <- proc.time()

    timetaken <- end.time - start.time
    list(opt_value = opt_value, SE = SE, time = timetaken[3])
}
```

Problem A – C

```
require(compiler)
enableJIT(3)

library(pracma)
MonteCarloGeoAsian <- function(isCall, S0, K, T, sig,div,r,Nt,M){
```

```r
    dt <- T/Nt
    nudt <- (r-div-0.5*sig^2)*dt
    sigsdt <- sig*sqrt(dt)
    lns <- log(S0)
    sum_OT <- 0
    sum_OT2 <- 0
    start.time <- proc.time()

    for (j in 1:M){
        w <- rnorm(Nt)
        lnSt <- lns
        sum_ST = vector("numeric", length = Nt)
        for(i in 1:Nt){
            lnSt <- lnSt + nudt + sigsdt*w[i]
            St <- exp(lnSt)
            sum_ST[i] = St
        }
        OT <- ifelse(isCall, max(0, (geomean(sum_ST)) -K), max(0,K-(geomean(sum_ST))))
        sum_OT <- sum_OT+OT
        sum_OT2 <- sum_OT2+OT*OT
    }
    opt_value <- sum_OT/M*exp(-r*T)
    SD <- sqrt((sum_OT2 - sum_OT*sum_OT/M)*exp(-2*r*T)/(M-1))
    SE <- SD/sqrt(M)

    end.time <- proc.time()

    timetaken <- end.time - start.time
    list(opt_value = opt_value, SE = SE, time = timetaken[3])
}
```

Problem A – D

```r
MCCVAsianOpt <- function(S0, sigma, K, r, Tm, Nt, M1, M2, type){
    library(pracma)
    start.time <- proc.time()

    dB <-matrix(rnorm(Nt*M1), nrow = Nt, ncol = M1)
    dt <- Tm/Nt

    k <- r - (sigma^2)*0.5
    deterministic <- k*dt
    stochastic <- sigma*sqrt(dt)*dB
    exp.path <- rbind(repmat(S0, 1, M1), exp(deterministic + stochastic))

    rm(list=ls(.GlobalEnv)[grep(deparse(substitute("dB")), ls(.GlobalEnv))],envir=.GlobalEnv)
    rm(list=ls(.GlobalEnv)[grep(deparse(substitute("deterministic")), ls(.GlobalEnv))],envir=.GlobalEnv)
    rm(list=ls(.GlobalEnv)[grep(deparse(substitute("stochastic")), ls(.GlobalEnv))],envir=.GlobalEnv)

    paths <- apply(exp.path, 2, FUN = cumprod)
    rm(list=ls(.GlobalEnv)[grep(deparse(substitute("exp.path")), ls(.GlobalEnv))],envir=.GlobalEnv)

    divisor <- 1/(Nt+1)
    DF <- exp(-r*Tm)

    geoExact <- geoAsianOpt(S0, sigma, K, r, Tm, Nt, type)
    geoCallPrices <- matrix(0, ncol = 1, nrow = M1)
    geoPutPrices <- matrix(0, ncol = 1, nrow = M1)
    ariCallPrices <- matrix(0, ncol = 1, nrow = M1)
    ariPutPrices <- matrix(0, ncol = 1, nrow = M1)

    #b and c
    for(i in 1:M1){
        pathVector <- paths[,i]
        avgPathPrice <- sum(pathVector)*divisor
```

```r
        if(type == 0){
            geoCallPrices[i] <- DF*max(geomean(pathVector) - K, 0)
            ariCallPrices[i] <- DF*max(avgPathPrice - K, 0)
        }else{
            geoPutPrices[i] <- DF*max(K - geomean(pathVector), 0)
            ariPutPrices[i] <- DF*max(K - avgPathPrice, 0)
        }
    }

    if(type == 0){
        geoCallOtp <- sum(geoCallPrices)/M1
        ariCallOtp <- sum(ariCallPrices)/M1
    }else{
        geoPutOtp <- sum(geoPutPrices)/M1
        ariPutOtp <- sum(ariPutPrices)/M1
    }

    if(type == 0){
        b <- cov(geoCallPrices, ariCallPrices)/var(geoCallPrices)
    }else{
        b <- cov(geoPutPrices, ariPutPrices)/var(geoPutPrices)
    }

    end.time <- proc.time()
    timetaken1 <- end.time - start.time

    start.time <- proc.time()

    rm(list=ls(.GlobalEnv)[grep(deparse(substitute("geoCallPrices")), ls(.GlobalEnv))], envir=.GlobalEnv)
    rm(list=ls(.GlobalEnv)[grep(deparse(substitute("geoPutPrices")), ls(.GlobalEnv))], envir=.GlobalEnv)
    rm(list=ls(.GlobalEnv)[grep(deparse(substitute("ariCallPrices")), ls(.GlobalEnv))], envir=.GlobalEnv)
    rm(list=ls(.GlobalEnv)[grep(deparse(substitute("ariPutPrices")), ls(.GlobalEnv))], envir=.GlobalEnv)

    controlVars <- matrix(0, ncol = 1, nrow = M2)

    #d
    for(i in 1:M2){
        pathVector <- paths[,i]
        avgPathPrice <- sum(pathVector)*divisor
        if(type == 0){
            geoCallPrice <- DF*max(geomean(pathVector) - K, 0)
            ariCallPrice <- DF*max(avgPathPrice - K, 0)
            controlVars[i] <- ariCallPrice - b*(geoCallPrice - geoExact)
        }else{
            geoPutPrice <- DF*max(K - geomean(pathVector), 0)
            ariPutPrice <- DF*max(K - avgPathPrice, 0)
            controlVars[i] <- ariPutPrice - b*(geoPutPrice - geoExact)
        }
    }

    price <- mean(controlVars)

    error <- sd(controlVars)/sqrt(M2)

    end.time <- proc.time()
    timetaken2 <- end.time - start.time

    rm(list=ls(.GlobalEnv)[grep(deparse(substitute("controlVars")), ls(.GlobalEnv))],envir=.GlobalEnv)

    if(type == 0){
        return(list(geo.Price = geoCallOtp, ari.Price = ariCallOtp, time1 = as.numeric(timetaken1[3]), b = 
as.numeric(b), CV.price = price, se.error = error, time2 = as.numeric(timetaken2[3]), total.time = as.nume
ric(timetaken1[3])+as.numeric(timetaken2[3])))
    }else{
        return(list(geo.Price = geoPutOtp, ari.Price = ariPutOtp, time1 = as.numeric(timetaken1[3]), b = a
s.numeric(b), CV.price = price, se.error = error, time2 = as.numeric(timetaken2[3]), total.time = as.numer
ic(timetaken1[3])+as.numeric(timetaken2[3])))
```

```
    }
}
```

Problem A – BONUS

```
setwd("C:/Users/nloychin/Desktop/New folder")
BB.data <- read.csv("IBM Asian Opt.csv", header =TRUE)

asian.opt <- matrix(0, nrow = 30, ncol = 1)
for(i in 1:30){
    asian.opt[i] <- MCCVAsianOpt(S0=144.15, sigma=BB.data[i,2], K=BB.data[i,1], r=0.017, Tm=BB.data[i,3],
Nt=BB.data[i,3]*252, M1=100000, M2=100000, type=0)$CV.price
}

table.compare <- NULL

for(i in c(0,5,10,15,20,25)){
    table.compare <- rbind(table.compare, asian.opt[i+(1:5)])
}
colnames(table.compare) <- c("1Y", "2Y", "3Y", "4Y", "5Y")
row.names(table.compare) <- c("K:144.15", "K:149.15", "K:154.15", "K:159.15", "K:164.15", "K:169.15")
table.compare
```

Problem B – 1

```
setwd("C:/Users/nloychin/Desktop/New folder")
data <- read.csv("sample_data.csv")
library(Sim.DiffProc)

## Package 'Sim.DiffProc', version 4.0
## browseVignettes('Sim.DiffProc') for more informations.

ModelSelection <- function(dataset, model = c("euler", "ozaki", "kessler")){

    best.model <- NULL

    for(i in 1:ncol(dataset)){
        data <- ts(dataset[,i],start = 0,frequency = 365)

        Tst <- NULL
        for(j in model){
            fx1 <- expression(theta[1]*x)
            gx1 <- expression(theta[2]*x^theta[3])
            model1 <- fitsde(data=data,drift=fx1,diffusion=gx1, start=list(theta1=1,theta2=1,theta3=1),pml
e=j)

            fx2 <- expression(theta[1]+theta[2]*x)
            gx2 <- expression(theta[3]*x^(theta[4]))
            model2 <- fitsde(data=data,drift=fx2,diffusion=gx2, start=list(theta1=1,theta2=1,theta3=1,thet
a4=1),pmle=j)

            fx3 <- expression(theta[1]+theta[2]*x)
            gx3 <- expression(theta[3]*sqrt(x))
            model3 <- fitsde(data=data,drift=fx3,diffusion=gx3, start=list(theta1=1,theta2=1,theta3=1),pml
e=j)

            fx4 <- expression(theta[1])
            gx4 <- expression(theta[2]^theta[3])
            model4 <- fitsde(data=data,drift=fx4,diffusion=gx4, start=list(theta1=1,theta2=1,theta3=1),pml
e=j)

            fx5 <- expression(theta[1]*x)
            gx5 <- expression(theta[2]+theta[3]*x^theta[4])
            model5 <- fitsde(data=data,drift=fx5,diffusion=gx5, start=list(theta1=1,theta2=1,theta3=1,thet
a4=1),pmle=j)

            AIC <- c(AIC(model1),AIC(model2),AIC(model3),AIC(model4),AIC(model5))
            tst <- data.frame(AIC,row.names=paste(c("model1","model2","model3","model4","model5"), ".", j,
sep = ""))
            Tst <- rbind(Tst, tst)
            print(Tst)
        }
        best <- rownames(Tst)[which.min(Tst[,1])]
        best.model <- rbind(best.model, best)
    }

    best.model <- data.frame(best.model)
    colnames(best.model) <- "best.model"
    row.names(best.model) <- c("stock1", "stock2", "stock3", "stock4", "stock5")

    return (best.model)
}
best.model <- ModelSelection(data)
```

Problem B – 2

```
EstimateParameter <- function(dataset, no, model=c("euler", "ozaki", "shoji", "kessler")){

    table <- NULL
    data1 <- ts(dataset,start = 1,frequency = 365)

    if(no == 1){
        for(j in model){
            fx1 <- expression(theta[1]*x)
            gx1 <- expression(theta[2]*x^theta[3])
            model1 <- fitsde(data=data1,drift=fx1,diffusion=gx1, start=list(theta1=1,theta2=1,theta3=1),pm
le=j)
            result <- data.frame(method = j, t(model1$coef), AIC = AIC(model1), BIC = BIC(model1), LogLik
= logLik(model1))
            table <- rbind(table, result)
        }
    }else if(no == 2){
        for(j in model){
            fx2 <- expression(theta[1]+theta[2]*x)
            gx2 <- expression(theta[3]*x^(theta[4]))
            model2 <- fitsde(data=data1,drift=fx2,diffusion=gx2, start=list(theta1=1,theta2=1,theta3=1,the
ta4=1),pmle=j)
            result <- data.frame(method = j, t(model2$coef), AIC = AIC(model2), BIC = BIC(model2), LogLik
= logLik(model2))
            table <- rbind(table, result)
        }
    }else if(no == 3){
        for(j in model){
            fx3 <- expression(theta[1]+theta[2]*x)
            gx3 <- expression(theta[3]*sqrt(x))
            model3 <- fitsde(data=data1,drift=fx3,diffusion=gx3, start=list(theta1=1,theta2=1,theta3=1),pm
le=j)
            result <- data.frame(method = j, t(model3$coef), AIC = AIC(model3), BIC = BIC(model3), LogLik
= logLik(model3))
            table <- rbind(table, result)
        }
    }else if(no == 4){
        for(j in model){

            fx4 <- expression(theta[1])
            gx4 <- expression(theta[2]^theta[3])
            model4 <- fitsde(data=data1,drift=fx4,diffusion=gx4, start=list(theta1=1,theta2=1,theta3=1),pm
le=j)
            result <- data.frame(method = j, t(model4$coef), AIC = AIC(model4), BIC = BIC(model4), LogLik
= logLik(model4))
            table <- rbind(table, result)
        }

    }else if(no == 5){
        for(j in model){
            fx5 <- expression(theta[1]*x)
            gx5 <- expression(theta[2]+theta[3]*x^theta[4])
            model5 <- fitsde(data=data1,drift=fx5,diffusion=gx5, start=list(theta1=1,theta2=1,theta3=1,the
ta4=1),pmle=j)
            result <- data.frame(method = j, t(model5$coef), AIC = AIC(model5), BIC = BIC(model5), LogLik
= logLik(model5))
            table <- rbind(table, result)
        }

    }
    return(table)
}
```

Problem C – 1

```
library(quantmod)
stockData <- new.env()
lookup.symb = c("MMM", "AXP", "AAPL", "BA", "CAT", "CVX", "CSCO", "KO", "DIS", "DWDP", "XOM",
                "GE", "GS", "HD", "IBM", "INTC", "JNJ", "JPM", "MCD", "MRK", "MSFT", "NKE", "PFE",
                "PG", "TRV", "UTX", "UNH", "VZ", "V", "WMT")

getSymbols(lookup.symb, from="2013-05-09", env=stockData, src="yahoo")
##  [1] "MMM"  "AXP"  "AAPL" "BA"   "CAT"  "CVX"  "CSCO" "KO"   "DIS"  "DWDP"
## [11] "XOM"  "GE"   "GS"   "HD"   "IBM"  "INTC" "JNJ"  "JPM"  "MCD"  "MRK"
## [21] "MSFT" "NKE"  "PFE"  "PG"   "TRV"  "UTX"  "UNH"  "VZ"   "V"    "WMT"

ReturnMatrix <- NULL

for(i in 1:length(lookup.symb)){
    tmp <- get(lookup.symb[i], pos=stockData)   # get data from stockData environment
    ReturnMatrix=cbind(ReturnMatrix,   dailyReturn(tmp)    )
    colnames(ReturnMatrix)[i]=lookup.symb[i]
}

r.bar <- colMeans(ReturnMatrix)
sigma.bar <- apply(ReturnMatrix, 2, sd)

library(pracma)
```

Problem C – 4

```
R.vec <- ReturnMatrix
sigma.vec <- sqrt(diag(cov(ReturnMatrix)))
V.vec <- eigen.return$vectors[,1]

F1 <- as.matrix(R.vec)%*%(eigen.return$vectors[,1]/sigma.vec/sqrt(eigen.return$values[1]))
sd.F1 <- sd(F1)
mean.F1 <- mean(F1)
```

Problem C – 6

```
F2 <- as.matrix(R.vec)%*%(eigen.return$vectors[,2]/sigma.vec/sqrt(eigen.return$values[2]))
F3 <- as.matrix(R.vec)%*%(eigen.return$vectors[,3]/sigma.vec/sqrt(eigen.return$values[3]))
F4 <- as.matrix(R.vec)%*%(eigen.return$vectors[,4]/sigma.vec/sqrt(eigen.return$values[4]))
F5 <- as.matrix(R.vec)%*%(eigen.return$vectors[,5]/sigma.vec/sqrt(eigen.return$values[5]))

return30.std <- apply(ReturnMatrix, 2, FUN = function(x){(x - mean(x))/sd(x)})
r30.bar <- colMeans(ReturnMatrix)
sigma30.bar <- apply(ReturnMatrix, 2, sd)

beta30 <- NULL

for(i in 1:length(lookup.symb)){
    lm.30 <- lm(return30.std[,i] ~ -1+F1+F2+F3+F4+F5)
    beta30 <- cbind(beta30, summary(lm.30)$coefficients[c(1:5)])
    colnames(beta30)[i] <- lookup.symb[i]
}
row.names(beta30) <- c('betaF1', 'betaF2', 'betaF3', 'betaF4', 'betaF5')
beta30 <- t(beta30)

portf <- NULL
port <- vector("numeric", length = 30)
for(i in 1:100000){
    for(j in 1:30){
        R <- matrix(r30.bar[j], nrow = 10, ncol = 1) + sigma30.bar[j]*(matrix(rt(10*5, df=3.5), nrow = 10,
```

```
ncol = 5)%*%(beta30[j,]))+
            sigma30.bar[j]*sqrt(1-sum(beta30[j,]^2))*matrix(rt(10, df=3.5), nrow = 10, ncol = 1)
        port[j] <- (1+R[1])*(1+R[2])*(1+R[3])*(1+R[4])*(1+R[5])*(1+R[6])*(1+R[7])*(1+R[8])*(1+R[9])*(1+R[1
0])-1
    }
    portf <- rbind(portf, sum(0.0333*port))
}
hist(sort(portf))
```