621_Homework 4

Liting Hu

Contents

Question 1

1

```r
require(Sim.DiffProc)
setwd("/Users/apple/Desktop/621/HW4")
sample_data <- read.csv("sample_data.csv") # Read data
sample_data <- ts(sample_data, deltat=1/365) # dt = 1/365
head(sample_data)
```

```
##         stock1    stock2    stock3    stock4   stock5
## [1,] 100.00000 100.00000 100.00000 100.00000 100.0000
## [2,] 100.02068  99.99415 100.04909  99.92849 100.1941
## [3,] 100.02660 100.01964  99.94416  99.92583 100.4359
## [4,] 100.00023 100.07849  99.94365  99.96918 100.4597
## [5,] 100.00754 100.13089  99.87517  99.93167 100.4614
## [6,]  99.96268 100.15621  99.91428  99.96727 100.3518
```

```r
model.match <- as.data.frame(matrix(nrow = 5, ncol = 2))
model.match[, 1] <- colnames(sample_data)
best.n <- c()
for (i in 1:5) {
    # model 1
    fx1 <- expression( theta[1]*x )
    gx1 <- expression( theta[2]*x^theta[3] )
    mod1 <- fitsde(data=sample_data[, i], drift=fx1, diffusion=gx1,
                   start = list(theta1=1, theta2=1, theta3=1), pmle="euler")

    # model 2
    fx2 <- expression( theta[1]+theta[2]*x )
    gx2 <- expression( theta[3]*x^theta[4] )
    mod2 <- fitsde(data=sample_data[, i], drift=fx2, diffusion=gx2,
                   start = list(theta1=1,theta2=1,theta3=1,theta4=1), pmle="euler")
```

```r
    # model 3
    fx3 <- expression( theta[1]+theta[2]*x )
    gx3 <- expression( theta[3]*sqrt(x) )
    mod3 <- fitsde(data=sample_data[, i], drift=fx3, diffusion=gx3,
                   start = list(theta1=1,theta2=1,theta3=1), pmle="euler")

    # model 4
    fx4 <- expression( theta[1] )
    gx4 <- expression( theta[2]*x^theta[3] )
    mod4 <- fitsde(data = sample_data[, i], drift=fx4, diffusion=gx4,
                   start = list(theta1=1,theta2=1,theta3=1), pmle="euler")

    # model 5
    fx5 <- expression( theta[1]*x )
    gx5 <- expression( theta[2]+theta[3]*x^theta[4] )
    mod5 <- fitsde(data=sample_data[, i], drift=fx5, diffusion=gx5,
                   start = list(theta1=1,theta2=1,theta3=1,theta4=1), pmle="euler")

    # Computes AIC
    AIC <- c(AIC(mod1),AIC(mod2),AIC(mod3),AIC(mod4),AIC(mod5))
    k <- which.min(AIC) # Choose the minimum
    best.n[i] <- k
    best <- paste0("model", k)
    model.match[i, 2] <- best
}
```

```r
knitr::kable(model.match, caption = "Model match")
```

Table 1: Model match

V1	V2
stock1	model1
stock2	model1
stock3	model1
stock4	model5
stock5	model1
The best	models are chosen as Model 1,1,1,5,1 respectively based on AIC.

2

```r
fx <- list(fx1, fx2, fx3, fx4, fx5)
gx <- list(gx1, gx2, gx3, gx4, gx5)
pmle <- eval(formals(fitsde.default)$pmle)

for (j in 1:5) {
    fitres <- lapply(1:4, function(i) fitsde(data = sample_data[, j], drift=fx[[best.n[j]]], diffusion=g
                                    pmle=pmle[i], start=list(theta1=1,theta2=1,theta3=1,theta4=
    Coef <- data.frame(do.call("cbind", lapply(1:4, function(i) coef(fitres[[i]]))))
    Info <- data.frame(do.call("rbind", lapply(1:4, function(i) logLik(fitres[[i]]))),
                       do.call("rbind", lapply(1:4, function(i) AIC(fitres[[i]]))),
                       do.call("rbind", lapply(1:4, function(i) BIC(fitres[[i]]))))
    Info <- t(Info)
```

```r
    colnames(Coef) <- pmle
    colnames(Info) <- pmle
    rownames(Info) <- c("logLic", "AIC", "BIC")
    result <- rbind(Coef, Info)
    print(paste0("stock", j))
    print(result)
}
```

```
## [1] "stock1"
##                euler    kessler         ozaki         shoji
## theta1  8.062477e-03  0.6357041  8.076304e-03  8.076296e-03
## theta2  4.298704e-02  0.4524182  4.348005e-02  4.348007e-02
## theta3  5.972266e-01 -1.2589008  5.952456e-01  5.952455e-01
## theta4  1.000000e+00  1.0000000  1.000000e+00  1.000000e+00
## logLic  1.281996e+05  0.0000000  1.281988e+05  1.281988e+05
## AIC    -2.563912e+05  8.0000000 -2.563895e+05 -2.563895e+05
## BIC    -2.563762e+05 23.0258709 -2.563745e+05 -2.563745e+05
## [1] "stock2"
##                euler       kessler         ozaki         shoji
## theta1  6.693138e-03  7.186735e-03  6.767031e-03  6.731218e-03
## theta2  3.173901e-02  3.107624e-02  3.199115e-02  3.198999e-02
## theta3  7.902994e-01  7.945091e-01  7.891739e-01  7.891783e-01
## theta4  1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00
## logLic  6.480208e+04  6.480452e+04  6.480113e+04  6.480115e+04
## AIC    -1.295962e+05 -1.296010e+05 -1.295943e+05 -1.295943e+05
## BIC    -1.295811e+05 -1.295860e+05 -1.295792e+05 -1.295793e+05
## [1] "stock3"
##            euler    kessler         ozaki         shoji
## theta1 -6.447688  0.6908933  7.585487e-03  7.720217e-03
## theta2 -1.240025  0.4878146 -1.121560e-02 -1.128753e-02
## theta3 -1.037941 -1.1314202  1.092039e+00  1.089914e+00
## theta4  1.000000  1.0000000  1.000000e+00  1.000000e+00
## logLic  0.000000  0.0000000 -1.499991e+04 -1.501172e+04
## AIC     8.000000  8.0000000  3.000783e+04  3.003145e+04
## BIC    23.025871 23.0258709  3.002285e+04  3.004648e+04
## [1] "stock4"
##                euler       kessler         ozaki         shoji
## theta1  3.501537e-03  3.620323e-03  3.547551e-03  3.499592e-03
## theta2 -3.106228e-01 -3.392665e-01 -3.626354e-01 -3.642739e-01
## theta3  8.066213e-02  8.523160e-02  8.853891e-02  8.862773e-02
## theta4  6.962709e-01  6.874293e-01  6.816353e-01  6.816288e-01
## logLic  6.525241e+04  6.525224e+04  6.525221e+04  6.525222e+04
## AIC    -1.304968e+05 -1.304965e+05 -1.304964e+05 -1.304964e+05
## BIC    -1.304818e+05 -1.304815e+05 -1.304814e+05 -1.304814e+05
## [1] "stock5"
##                euler       kessler         ozaki         shoji
## theta1  6.347047e-03  5.134631e-03  6.294413e-03  6.336781e-03
## theta2  4.494081e-02  4.444700e-02  4.513493e-02  4.507670e-02
## theta3  7.893863e-01  7.911148e-01  7.884441e-01  7.888489e-01
## theta4  1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00
## logLic  2.721819e+04  2.721769e+04  2.721773e+04  2.721795e+04
## AIC    -5.442838e+04 -5.442737e+04 -5.442746e+04 -5.442790e+04
## BIC    -5.441336e+04 -5.441234e+04 -5.441243e+04 -5.441287e+04
```

For stock 1, the minimum AIC and BIC appear in euler method. Those calculated by kessler method are much larger (8 >> -2.563912e+05, 23.0258709 >> -2.563762e+05).

For stock 2, the minimum AIC and BIC appear in kessler method. And euler's AIC and BIC are also less than ozaki's and shoji's.

For stock 3, the minimum AICs and BICs appear in euler and kessler methods which are much smaller than ozaki's and shoji's.

For stock 4, the minimum AIC and BIC appear in euler method. And kessler's AIC and BIC are also less than ozaki's and shoji's.

For stock 5, the minimum AIC and BIC appear in euler method. And kessler's AIC and BIC are also less than ozaki's and shoji's. (similar situation as stock 4)

3

For my opinion, the Euler method and Kessler method provide best estimates.

Question 2

```
library(xlsx)
swaptions <- read.xlsx("2017_2_15_mid.xlsx", sheetName = "Quotes") # Read data
head(swaptions)
```

```
##   Expiry    NA.  X1Yr  X2Yr  X3Yr  X4Yr  X5Yr  X7Yr X10Yr X12Yr X15Yr
## 1    1Mo    Vol 28.79 44.15 41.98 65.02 70.63 75.92 72.92 73.68 73.57
## 2   <NA> Strike  1.37  1.62  1.83  1.98  2.10  2.27  2.44  2.52  2.60
## 3    3Mo    Vol 42.91 53.08 65.36 74.23 68.76 76.72 81.33 80.44 77.75
## 4   <NA> Strike  1.47  1.71  1.90  2.04  2.15  2.32  2.48  2.55  2.62
## 5    6Mo    Vol 42.71 58.20 67.08 75.01 80.60 83.30 84.40 83.64 81.40
## 6   <NA> Strike  1.61  1.82  2.00  2.12  2.22  2.37  2.52  2.59  2.65
##   X20Yr X25Yr X30Yr
## 1 73.12 72.41 71.92
## 2  2.66  2.68  2.68
## 3 77.47 77.17 76.83
## 4  2.68  2.69  2.69
## 5 84.39 81.52 81.19
## 6  2.70  2.71  2.71
```

```
column <- 5 # choose 5th maturity which is 5 years
dim(swaptions)
```

```
## [1] 38 14
```

```
Vol <- swaptions[seq(1, 38, by=2), 7]/100 # At-the-money volatility
Strike <- swaptions[seq(2, 38, by=2), 7]/100 # Strike price K
expiry <- swaptions[seq(1, 38, by=2), 1] # expiration date

temp <- gregexpr("[0-9]+", expiry)
expiry <- as.numeric(unlist(regmatches(expiry, temp)))
expiry[1:4] <- expiry[1:4]/12 # Expiry date for swaptions
```

1

```r
SABRVol<-function(alpha,beta,rho,nu,Tm,f){
    # All the contracts are at the money, so f=K
        Term1 <- alpha/f^(1-beta)
        Term2 <- (1-beta)^2/24*alpha^2/f^(2-2*beta)+rho*beta*nu*alpha/4/f^(1-beta)+(2-3*rho^2)*nu^2/24
        ans <- Term1*(1 + Term2*Tm)
        return(ans)
}

optimization <- function(beta) {
    init.values = c(0.1, 0, 0.1)
    lower.bound = c(0.0001, -1, 0.0001) # -1 < rou < 1
    upper.bound = c(Inf, 1, Inf)

    objective.f <- function(parm,beta) {
        # Objective function
        # to search the minimum sum of (sigma_mkt - sigma_B)^2
        alpha <- parm[1]
        rho <- parm[2]
        nu <- parm[3]
        Sigma_B <- SABRVol(alpha,beta,rho,nu,expiry,Strike)
        return(sum((Vol-Sigma_B)^2))
    }

    # PORT optimization routine
    opt<-nlminb(start=init.values,
                objective = objective.f,
                lower = lower.bound,
                upper = upper.bound,
                beta=beta)

    parms <-opt$par
    obj <- opt$objective
    ans <- t(as.data.frame(c(beta, parms, obj)))
    colnames(ans) <- c("beta", "alpha", "rho", "nu", "objective")
    rownames(ans) <- NULL
    return(ans)

}
re1 <- optimization(0.5)
```

```r
knitr::kable(re1, caption = "Beta=0.5")
```

Table 2: Beta=0.5

beta	alpha	rho	nu	objective
0.5	0.1278108	-0.7087668	3.214501	0.1607094

2

```r
re2 <- optimization(0.7)
re3 <- optimization(0.4)
```

```r
knitr::kable(re2, caption = "Beta=0.7")
```

Table 3: Beta=0.7

beta	alpha	rho	nu	objective
0.7	0.2701053	-0.6340573	2.601383	0.1253063

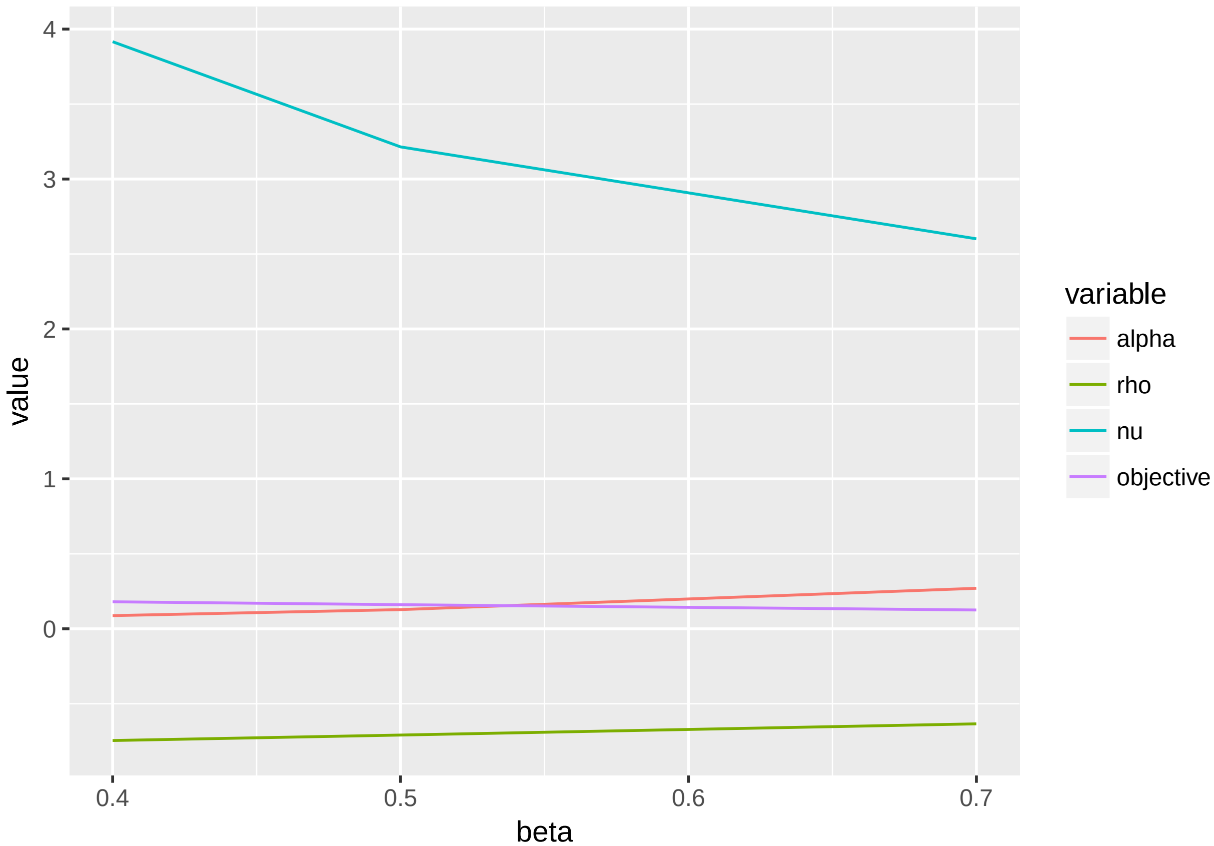
```r
knitr::kable(re3, caption = "Beta=0.4")
```

Table 4: Beta=0.4

beta	alpha	rho	nu	objective
0.4	0.0878855	-0.7454043	3.915984	0.1800368

3

```r
library(ggplot2)
library(reshape2)
res <- as.data.frame(rbind(re1, re2, re3))

res.long <- melt(res, id="beta")
ggplot(data=res.long, aes(x=beta, y=value, color=variable)) + geom_line()
```



From this plot, we can see that as beta (3 values) grows larger, the objective function decreases. While in the mean time, alpha and rho is getting larger. Nu is decreasing.

4

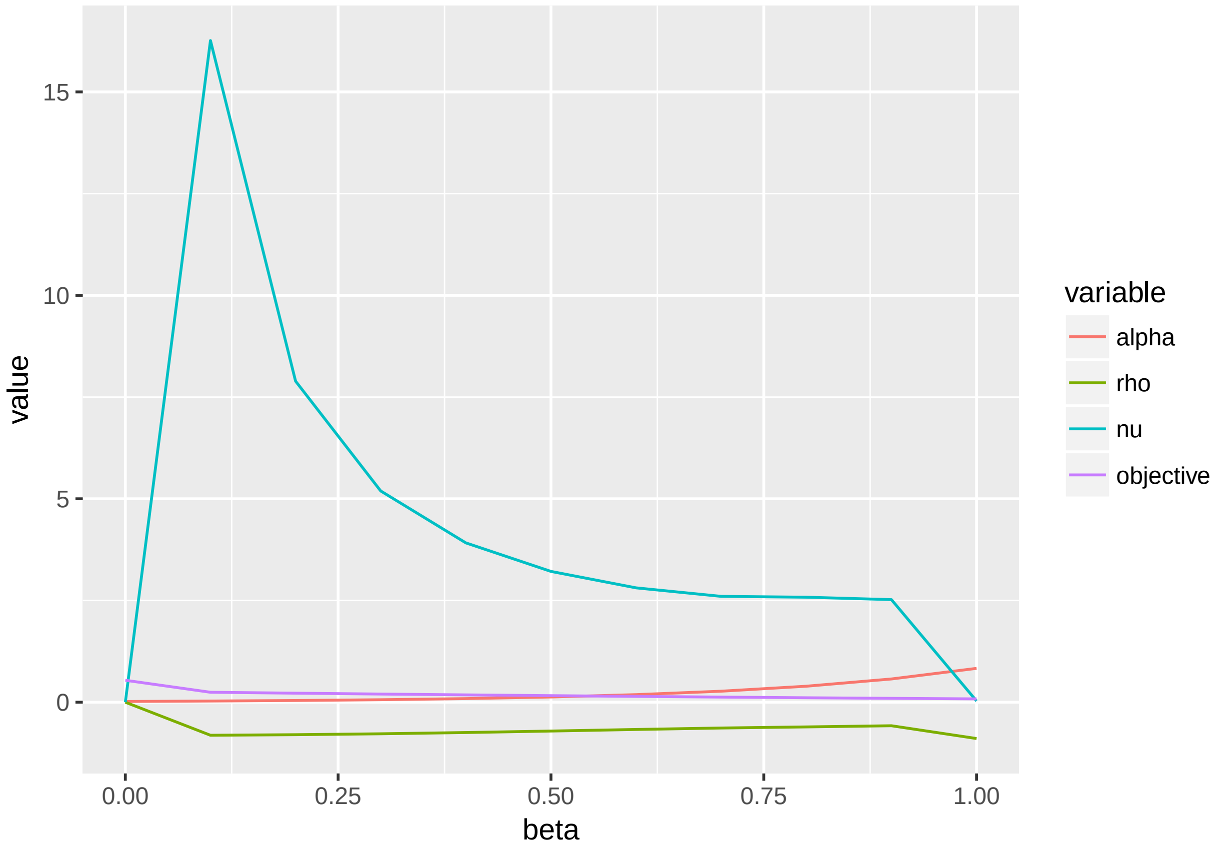
```
betas <- seq(0, 1, by=0.1)
opts <- optimization(0)
for (i in betas) {
    a <- optimization(i)
    opts <- rbind(opts, a)
}

opts <- as.data.frame(opts[2:12, ])
```

```
knitr::kable(opts)
```

beta	alpha	rho	nu	objective
0.0	0.0175400	0.0000000	0.0001000	0.5388539
0.1	0.0285304	-0.8123266	16.2623907	0.2441848
0.2	0.0415228	-0.7990391	7.8908936	0.2218027
0.3	0.0604166	-0.7763493	5.1921226	0.2004107
0.4	0.0878855	-0.7454043	3.9159839	0.1800368
0.5	0.1278108	-0.7087668	3.2145007	0.1607094
0.6	0.1858261	-0.6701024	2.8108362	0.1424566
0.7	0.2701053	-0.6340573	2.6013829	0.1253063
0.8	0.3925044	-0.6075992	2.5809287	0.1092863

beta	alpha	rho	nu	objective
0.9	0.5711641	-0.5787124	2.5210188	0.0945451
1.0	0.8313951	-0.8933951	0.0301414	0.0816091

```
opts.long <- melt(opts, id="beta")
ggplot(data=opts.long, aes(x=beta, y=value, color=variable)) + geom_line()
```

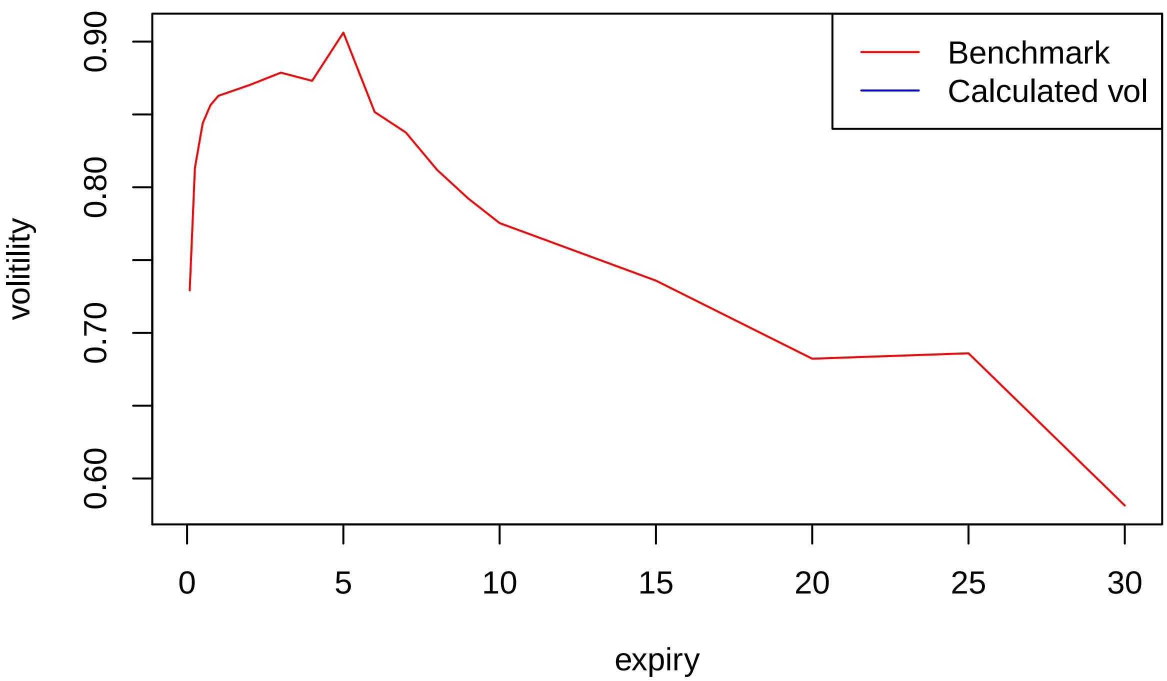


To be more precisely, in this section, let beta equal to 0, 0.1, 0.2, ... 1. Still, objective value is decreasing as beta growing. We can basically say that beta=1 gives me the best estimation.

5

```
beta <- 1
re5 <- optimization(beta)
alpha <- re3[2]
rho <- re3[3]
nu <- re3[4]

column <- 7 # choose 7th maturity which is 10 years
Vol2 <- swaptions[seq(1, 38, by=2), 9]/100 # At-the-money volatility
Strike2 <- swaptions[seq(2, 38, by=2), 9]/100 # Strike price K

VolB <- SABRVol(alpha,beta,rho,nu,expiry,Strike2)
plot(expiry, Vol2, type = "l", col = "red", ylab = "volitility")
lines(expiry, VolB, col = "blue")
```

```r
legend("topright", legend=c("Benchmark", "Calculated vol"),
       col=c("red", "blue"), lty=c(1,1))
```




```r
obj <- sum((VolB-Vol2)^2)
obj
```

```
## [1] 7.495555
```

Apply beta = 1 to swaps whose maturity is 10 years. The volitility calculated by SABR model is nearly a straight line. Given volitilities fluctuate around it. The SSE is 0.05352116 which is fairly small.

3

```r
S0=1
Tm=1
sigma=0.2
r=0.0075

alpha <- 1
N <- 10000
eta <- 50
lambda <- 2*pi/N/eta
a <- N*eta # the effective upper limit for the integration
b <- N*lambda/2 # log of strike prices range from -b to b

# for phi_T
f_phi<- function(u, s, mu, sigma) {
    # integrand
    ans <- 1/sqrt(2*pi*sigma^2)*exp(-(s-mu)^2/2/sigma^2)*exp(1i*u*s)
    return(ans)
}
```

```r
phi_u <- function(u, mu, sigma, a, N) {
    # Integral
    # By the Simpson's rule
    s <- seq(-a, a, length.out = N)
    h <- s[2] - s[1]
    wn <- rep(c(2*h/3, 4*h/3), floor(N/2))
    wn[1] <- h/3
    wn[N] <- h/3
    fn <- sapply(s, f_phi, u = u, mu = mu, sigma = sigma)
    ans <- sum(fn*wn)
    return(ans)
}

# Out the money
zeta_v <- function(v, r, Tm, mu, sigma, a, N) {
    ans <- exp(-r*Tm)*(1/(1 + 1i*v) - exp(r*Tm)/(1i*v) -
                           phi_u(v - 1i, mu, sigma, a, N)/(v^2 - 1i*v))
    return(ans)
}

gamma_v <- function(v, r, Tm, mu, sigma, alpha, a, N) {
    ans <- (zeta_v(v - 1i*alpha, r, Tm, mu, sigma, a, N)-
                zeta_v(v + 1i*alpha, r, Tm, mu, sigma, a, N))/2
    return(ans)
}

# In the money
psi_v <- function(v, r, Tm, mu, sigma, alpha, a, N) {
    ans <- (exp(-r*Tm)*phi_u(v - (alpha + 1)*1i, mu, sigma, a, N))/
        (alpha^2 + alpha - v^2 + 1i*(2*alpha + 1)*v)
    return(ans)
}


Call_FFT <- function(u, r, Tm, mu, sigma, alpha, a, N) {
    ku <- -b + lambda*(u-1)
    s <- seq(0.0001, a, length.out = N)
    h <- s[2] - s[1]
    wn <- rep(c(2*h/3, 4*h/3), floor(N/2))
    wn[1] <- h/3
    wn[N] <- h/3
    j <- seq(1, N)
    vj <- (j-1)*eta
    if (u < ((N+1)/2)) {
        f <- exp(-1i*2*pi/N*(j-1)*(u-1))*exp(1i*b*vj)*psi_v(vj, r, Tm, r, sigma, alpha, a, N)
        C <- exp(-alpha*k)/pi*sum(f*wn)
    } else {
        f <- exp(-1i*2*pi/N*(j-1)*(u-1))*exp(1i*b*vj)*gamma_v(vj, r, Tm, r, sigma, alpha, a, N)
        C <- sinh(alpha*k)/pi*sum(f*wn)
    }
    return(C)
}
u=60
```

```r
Call_FFT(u, r, Tm, r, sigma, alpha, a, N)
```

```
## [1] NaN+NaNi
```

```r
K <- (-b + (u-1)*lambda)^2
Option_BSM <- function(isCall = T, S0=1, K=1, Tm=1, sigma=0.2, r=0.0075, div=0) {
    d1 <- (log(S0/K) + (r - div + sigma^2/2)*Tm)/sigma/sqrt(Tm)
    d2 <- d1 - sigma*sqrt(Tm)
    if (isCall) {p <- S0*exp(-div*Tm)*pnorm(d1) - K*exp(-r*Tm)*pnorm(d2)}
    else {p <- K*exp(-r*Tm)*pnorm(-d2) - S0*exp(-div*Tm)*pnorm(-d1)}
    return(p)
}
Option_BSM(K=K)
```

```
## [1] 0.9961736
```