FE621 HW4 Bonus Part

Shihao Zhang

2018-12-05

Due to the large calculation of Monte Carlo Simulation inthe HW4, I have to seperate the HW4 orignal part and the bonus part to report in pdf. For the orignal problem part, please see FE621 HW4 Shihao Zhang Part1.

(BONUS 1) SABR parameter estimation

```
##question 1-----------------------------------------------------------------
#For this question, pick 2 yr matuarity
library('readxl')
setwd("C:\\Users\\fukaeri\\Desktop\\Stevens\\18FALL\\FE621\\HW")
mydataSABR<-read_excel("2017_2_15_mid.xlsx",col_names = TRUE)
#From equation(3) in paper,the at-the-money volatility sigma_ATM
Sigma_ATM<-function(alpha,beta,k,pho,v,t){
  term1<-((1-beta)^2)/24
  term2<-(alpha^2)/(k^(2-2*beta))
  term3<-(0.25*pho*beta*v*alpha)/(k^(1-beta))
  term4<-((2-3*pho^2)*(v^2))/24
  term5<-k^(1-beta)

  sig<-alpha*(1+(term1*term2+term3+term4)*t)/term5
  return(sig)
}
#Choose to use the 2rd yr data
Vol<-mydataSABR[seq(1,37,2),4]/100 #Volatility
K<-mydataSABR[seq(2,38,2),4]/100 #Strike Price

#Implement equation (5) in paper
f1<-function(x){
  sum=0
  for(i in 1:19){
    sum<-(Vol[i,1]-Sigma_ATM(x[1],0.5,K[i,1],x[2],x[3],2))^2+sum
  }
  return(sum)
}
#Apply Optimization function
library("nloptr")
#beta=0.5&out put the result
Beta <- 0.5
parameter_0.5<-bobyqa(c(2,0.3,0.5),f1)
SABR.parameter_0.5 <- c(parameter_0.5$par[1],Beta,parameter_0.5$par[2],
                        parameter_0.5$par[3],parameter_0.5$value)
names(SABR.parameter_0.5) <- c('alpha','beta', 'rho', 'nu','SSE')
SABR.parameter_0.5 <- as.data.frame(SABR.parameter_0.5)
SABR.parameter_0.5
```

```
##       SABR.parameter_0.5
```

```
## alpha          0.4032674
## beta           0.5000000
## rho           -0.7295207
## nu             2.0971079
## SSE            0.0905006
```

```
##question 2-------------------------------------------------------------
#Set beta=0.7&0.4 repeat part1
#beta=0.7
f2<-function(x){
  sum=0
  for(i in 1:19){
    sum<-(Vol[i,1]-Sigma_ATM(x[1],0.7,K[i,1],x[2],x[3],2))^2+sum
  }
  return(sum)
}
Beta <- 0.7
parameter_0.7<-bobyqa(c(2,0.3,0.5),f2)
SABR.parameter_0.7 <- c(parameter_0.7$par[1],Beta,parameter_0.7$par[2],
                        parameter_0.7$par[3],parameter_0.7$value)
names(SABR.parameter_0.7) <- c('alpha','beta', 'rho', 'nu','SSE')
SABR.parameter_0.7 <- as.data.frame(SABR.parameter_0.7)
SABR.parameter_0.7
```

```
##       SABR.parameter_0.7
## alpha         0.94751665
## beta          0.70000000
## rho          -0.58808483
## nu            1.76241928
## SSE           0.09484939
```

```
#beta=0.4
f3<-function(x){
  sum=0
  for(i in 1:19){
    sum<-(Vol[i,1]-Sigma_ATM(x[1],0.4,K[i,1],x[2],x[3],2))^2+sum
  }
  return(sum)
}
Beta <- 0.4
parameter_0.4<-bobyqa(c(2,0.3,0.5),f3)
SABR.parameter_0.4 <- c(parameter_0.4$par[1],Beta,parameter_0.4$par[2],
                        parameter_0.4$par[3],parameter_0.4$value)
names(SABR.parameter_0.4) <- c('alpha','beta', 'rho', 'nu','SSE')
SABR.parameter_0.4 <- as.data.frame(SABR.parameter_0.4)
SABR.parameter_0.4
```

```
##       SABR.parameter_0.4
## alpha         0.25565464
## beta          0.40000000
## rho          -0.75895157
## nu            2.84426229
## SSE           0.08889656
```

```
##question 3-------------------------------------------------------------
mycomparetable <- cbind(SABR.parameter_0.4,SABR.parameter_0.5,SABR.parameter_0.7)
```

```r
mycomparetable
```

```
##       SABR.parameter_0.4 SABR.parameter_0.5 SABR.parameter_0.7
## alpha         0.25565464          0.4032674         0.94751665
## beta          0.40000000          0.5000000         0.70000000
## rho          -0.75895157         -0.7295207        -0.58808483
## nu            2.84426229          2.0971079         1.76241928
## SSE           0.08889656          0.0905006         0.09484939
```

```r
#Comments:By comparsion, we notice that alpha increase when beta increase.
#Rho is decreasing when beta increasing(however it only change slightly)
#And nu is on opposition from the rho's direction
##question 4-------------------------------------------------------------
mycomparetable
```

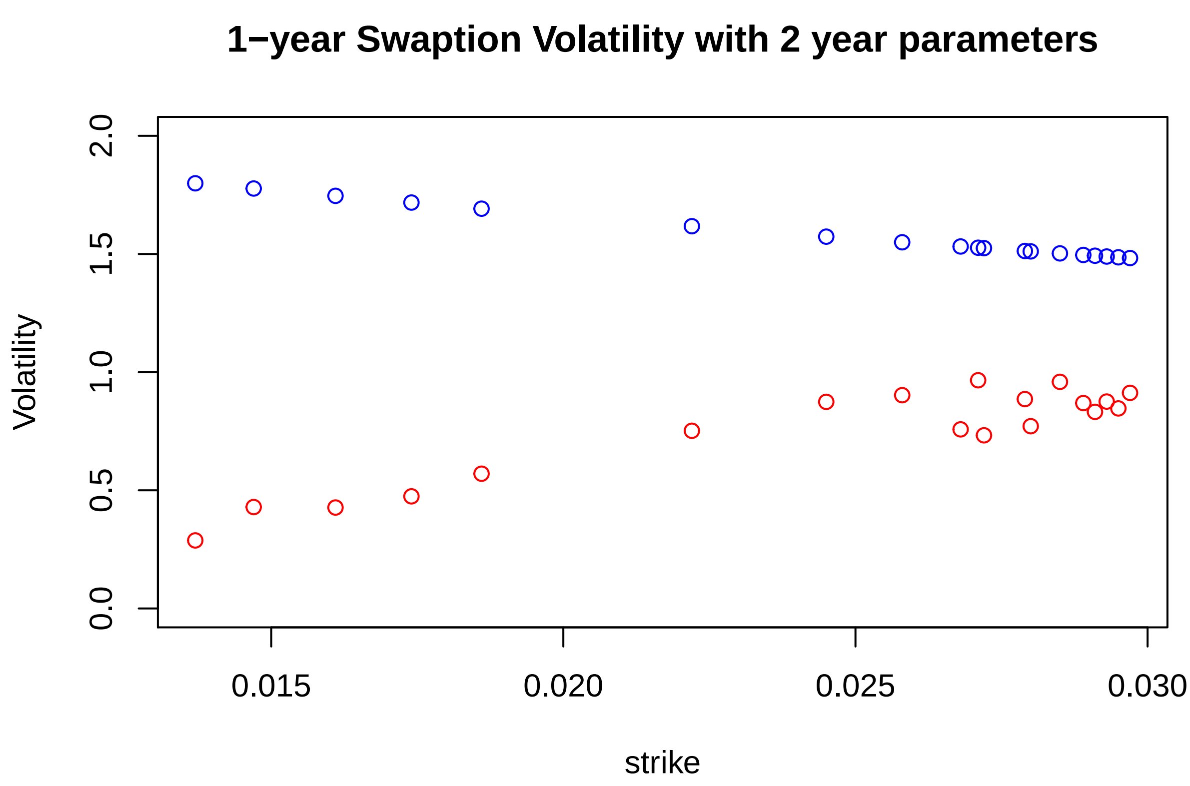

```
##       SABR.parameter_0.4 SABR.parameter_0.5 SABR.parameter_0.7
## alpha         0.25565464          0.4032674         0.94751665
## beta          0.40000000          0.5000000         0.70000000
## rho          -0.75895157         -0.7295207        -0.58808483
## nu            2.84426229          2.0971079         1.76241928
## SSE           0.08889656          0.0905006         0.09484939
```


```r
#Comments:Still by the comparsion table, the model gives us the best estimation
#when beta is 0.4. At this time, we obtain alpha=.2556546, rho=-0.7589516,
#nu=2.8442623, and the smallest SEE=0.0888966


##question 5-------------------------------------------------------------
alpha_best <- SABR.parameter_0.4[1,]
beta_best <- SABR.parameter_0.4[2,]
rho_best <- SABR.parameter_0.4[3,]
nu_best <- SABR.parameter_0.4[4,]
#Choose to use the 3rd yr data
K2 <- mydataSABR[seq(2,38,2), 3]/100
Vol2 <- mydataSABR[seq(1,38,2), 3]/100
vol_ATM <- matrix(NA,19,1)


for(i in 1:19){
  vol_ATM[i,1] <- Sigma_ATM(alpha_best,beta_best,K2[i,],rho_best,nu_best,2)
}
vol_atm <- matrix(NA,19,1)
for(i in 1:19){
  vol_atm[i,1] <- Sigma_ATM(alpha_best,beta_best,K2[i,],rho_best,nu_best,1)
}
compare2 <- as.matrix(cbind(Vol2 - vol_ATM,Vol2 - vol_atm))
colnames(compare2) <- c('1-year','2-year')
print(compare2)
```

```
##             1-year     2-year
##  [1,]  0.044022446 -1.5112639
##  [2,]  0.090564295 -1.3479725
##  [3,] -0.020313283 -1.3190045
##  [4,] -0.054638963 -1.2432149
##  [5,] -0.021092632 -1.1215444
```

```
## [6,]  0.027640105 -0.8657160
## [7,]  0.094224716 -0.6991171
## [8,]  0.097934943 -0.6469661
## [9,]  0.140437131 -0.5608209
## [10,]  0.049764912 -0.6267320
## [11,]  0.115292356 -0.5435827
## [12,]  0.020744184 -0.6268069
## [13,] -0.008360362 -0.6395267
## [14,]  0.055390084 -0.5704686
## [15,]  0.022747520 -0.6138025
## [16,] -0.018584802 -0.6605958
## [17,] -0.066223820 -0.7397291
## [18,] -0.094001054 -0.7920811
## [19,] -0.062928461 -0.7738684
```

```
plot(x=as.matrix(K2),y=vol_atm,col="blue",type="p",
     main="1-year Swaption Volatility with 2 year parameters",
     xlab="strike", ylab="Volatility", ylim = c(0, 2))
points(x=as.matrix(K2),y=as.matrix(Vol2),col="red")
```




```
#Comments:Estimate Volatility is blue points.
#And the real Volatility is red points.
#They converge when strike price is high.
```

(BONUS 2) Sim.DiffProc question

Comments: #1.When we apply Euler method to estimate the stochastic differential equations, the Euler scheme produces the discretization when delta t is approaching zero, and we have the increments (X[t+deltat]-X[t]) with certain mean(drift) and variance(diffusion). Then we can estimate the parameter by change the question into optimazing the log-likelihood, and we can select the optimization method by the argument(optim.method). #2.When we apply the Ozaki method, the diffusion term(sigma) is supposed to be constant. And we can transform general SDE with a constant diffusion coefficient using the Lamperti transform.

```
set.seed(1)
#Given the information
S0 <- 100
theta1 <- 1000
theta2 <- -10
theta3 <- 0.8
theta4 <- 0.5
dt <- 1/365
Tm <- 4
#Simulate the path
library(Sim.DiffProc)
```

```
## Package 'Sim.DiffProc', version 4.3
## browseVignettes('Sim.DiffProc') for more informations.
```

```
f <- expression((theta1+theta2*x))
g <- expression(theta3*x^theta4)
sim <- snssde1d(drift=f,diffusion=g,x0=S0,M=1,N=1460,Dt=dt)
mydata <- sim$X
#Estimation of model
fx <- expression(theta[1]+theta[2]*x) ##drift coefficient
gx <- expression(theta[3]*x^theta[4]) ##diffusion coefficient
#1.Euler method
fitmod_Euler <- fitsde(data=mydata,drift=fx,diffusion=gx,start=list(theta1=999,
                 theta2=10,theta3=1,theta4=1),pmle="euler")
coef_Euler <- coef(fitmod_Euler)
true <- true_value <- c(theta1,theta2,theta3,theta4) ##True parameters
bias_Euler <- true-coef(fitmod_Euler)
AIC_Euler <- AIC(fitmod_Euler)
#2.Ozaki method
fitmod_Ozaki <- fitsde(data=mydata,drift=fx,diffusion=gx,start=list(theta1=999,
                 theta2=10,theta3=1,theta4=1),pmle="ozaki")
coef_Ozaki <- coef(fitmod_Ozaki)
bias_Ozaki <- true-coef(fitmod_Ozaki)
AIC_Ozaki  <- AIC(fitmod_Ozaki)
#3.Shoji-Ozaki method
fitmod_Shoji <- fitsde(data=mydata,drift=fx,diffusion=gx,start=list(theta1=999,
                 theta2=10,theta3=1,theta4=1),pmle="shoji")
coef_Shoji <- coef(fitmod_Shoji)
bias_Shoji <- true-coef(fitmod_Shoji)
AIC_Shoji  <- AIC(fitmod_Shoji)
#4.Kessler method
fitmod_Kessler <- fitsde(data=mydata,drift=fx,diffusion=gx,start=list(theta1=999,
                 theta2=10,theta3=1,theta4=1),pmle="kessler")
coef_Kessler <- coef(fitmod_Kessler)
```

```r
bias_Kessler <- true-coef(fitmod_Kessler)
AIC_Kessler  <- AIC(fitmod_Kessler)
#Create Table and Report
#true value and estimated coef
myresult1 <- cbind(true_value,coef_Euler,coef_Ozaki,coef_Shoji,coef_Kessler)
myresult1
```

```
##        true_value  coef_Euler  coef_Ozaki  coef_Shoji coef_Kessler
## theta1     1000.0 998.7996332 998.8050204 998.8050133  998.9134885
## theta2      -10.0  -9.9990326  -9.9990220  -9.9990070    1.1276705
## theta3        0.8   0.8648526   0.8509538   0.8226366    0.5810937
## theta4        0.5   0.4892249   0.4927461   0.5030613   -0.3251940
```

```r
#Bias
myresult2 <- cbind(bias_Euler,bias_Ozaki,bias_Shoji,bias_Kessler)
myresult2
```

```
##           bias_Euler    bias_Ozaki    bias_Shoji bias_Kessler
## theta1  1.2003667907  1.1949795597  1.1949867098    1.0865115
## theta2 -0.0009673675 -0.0009779943 -0.0009930413  -11.1276705
## theta3 -0.0648525587 -0.0509537782 -0.0226365975    0.2189063
## theta4  0.0107750520  0.0072539410 -0.0030612974    0.8251940
```

```r
#AIC
#AIC deals with the trade-off between the goodness of fit
#of the model,AIC lower is preferred.
myresult3 <- cbind(AIC_Euler,AIC_Ozaki,AIC_Shoji,AIC_Kessler)
myresult3
```

```
##      AIC_Euler AIC_Ozaki AIC_Shoji AIC_Kessler
## [1,]  1689.626  1689.674  1689.671           8
```

```r
#Comments:The Euler,Ozaki,Shoji-Ozaki scheme all fit the process at pretty much same level,
#with almost identical parameter estimation, Bias and AIC.
#However, when it turn to Kessler scheme, the Kessler scheme might be the best for fitting
#the process with the lowest AIC(AIC=8), but the paramater bias for theta2 is high
#(and negative).
```