Bonus

Problem 4

(a)

```r
alpha=0.01
beta=0.1
gamma=0.02

sigma <- function(x){
  return(alpha*x^2+beta*x+gamma)
}

sigma.fun <- function(x){
  return(1/(alpha*x^2+beta*x+gamma))
}

s <- function(x0,x){
  integrate(sigma.fun,lower = x0,upper = x)$value
}

Q <- alpha*gamma/2 - beta^2/8

p <-function(t,x0,x){
  term1 = sigma.fun(x)/sqrt(2*pi*t)
  term2 = exp(-1/(2*t)*(integrate(sigma.fun,lower = x0,upper = x)$value^2)+Q*t)
  return(term1*sigma(x0)/sigma(x)*term2)
}
# Creat 3D plot
COLOR=c(rep("red",20),rep("blue",20),rep("green",20))
library(scatterplot3d)
t <- seq(1,2,0.02)
x <- seq(1,10,0.1)
p.value <- c()
# for(j in 1:100){
#   for(i in 1:100){
#     p.value <- c(p.value,p(t[j],0,x[i]))
#   }
# }
#scatterplot3d(t,x,p.value,main="3D plot",color = COLOR,pch=16,type = "p")
```

(b)

```r
delta.fun <- function(x){
  if(x==0)
    return(1)
  else
    return(0)
```

```r
}

Finite <- function(t,x0,x){
  n = 1000
  delta.x = 0.01
  p0 <- delta.fun(s(x0,x)-s(x0,x0))
  #Loop over time steps
  p.value = p0
  for(i in 1:n){
    delta.p = 0.5*sigma(x)^2*(p(t,x0,x+delta.x) - 2*p(t,x0,x) + p(t,x0,x-delta.x))/delta.x^2
    p.value = p.value + delta.p
  }
  return(p.value)
}
Finite(30/252,1.03,1)
```

```
## [1] -21169.93
```

(c)

```r
s <- function(x0,K){
  abs(integrate(sigma.fun,lower = x0,upper = K)$value)
}

C <- function(tau,K,x0){
  term1 = sigma(K)*sigma(x0)/(2*sqrt(-2*Q))
  term2 = exp(s(x0,K)*sqrt(-Q))*pnorm(-s(x0,K)/sqrt(2*tau) - sqrt(-2*Q*tau))
  term3 = exp(-s(x0,K)*sqrt(-Q))*pnorm(-s(x0,K)/sqrt(2*tau) + sqrt(-2*Q*tau))
  return(max(x0-K,0)+term1*(term2-term3))
}
C(30/252,1,1.03)
```

```
## [1] 0.02875156
```

```r
#calculate European call option price by BS model
library(RQuantLib)
EuropeanOption("call", underlying = 1, strike = 1,
                                            dividendYield = 0, riskFreeRate = 0,
                                            maturity = 30/252, volatility = 0.2)$value
```

```
## [1] 0.02756999
```

Problem 5

(a)

```r
Heston_Price <- function(params,S0,r,q,T,k){
  #input all parameters
  kappa = params[1];
  theta = params[2];
  sigma = params[3];
```

```
  V0    = params[4];
  rho   = params[5];
  lambda = params[6];

  i <- sqrt(-1+0i)
  kappa <- c(4,2,1)
  u1 <- 0.5
  u2 <- -0.5
  a <- kappa*theta
  b1 <- kappa+lambda-rho*sigma
  b2 <- kappa+lambda
  N = 1e+04
  up = 700

  psi <- function(Phi,b,u){
    d <- sqrt((rho*sigma*Phi*i-b)^2 - sigma^2*(2*u*Phi*i-Phi^2))
    g <- (b-rho*sigma*Phi*i+d)/(b-rho*sigma*Phi*i-d)
    C <- (r-q)*Phi*i*T+kappa*theta*((b-rho*sigma*Phi*i+d)*T -
                                    2*log((1-g*exp(d*T))/(1-g)))/sigma^2
    D <- (b-rho*sigma*Phi*i+d)/sigma^2 * ((1-exp(d*T))/(1-g*exp(d*T)))
    y <- exp( C+D*V0+i*Phi*log(S0))
    return(y)
  }

  expre <- function(Phi,b,u,k){
    y <- Re(exp(-i*Phi*log(k))*psi(Phi,b,u)/(i*Phi))
    return(y)
  }

  #use simpon method to calculate integral
  Simpon <- function(up,N,b,u,k){
    In <- 0
    h <- up/N
    for(j in 1:N+1){
      Phi <- j*h
      Phi1 <- (j-1)*h
      area <- (Phi - Phi1)/6 * (expre(Phi1,b,u,k) +
                                4*expre((Phi + Phi1)/2,b,u,k)+expre(Phi,b,u,k))
      In <- area + In
    }
    return(In)
  }

  #calculate option price by Hedson model
  simpon1 <- mean(Simpon(up,N,b1,u1,k))
  simpon2 <- mean(Simpon(up,N,b2,u2,k))
  P1 <- 1/2 + 1/pi * simpon1
  P2 <- 1/2 + 1/pi * simpon2
  price <- S0*P1 - k*exp(-(r-q)*T)*P2
  return(price)
}
Heston_Price(params=c(4,0.1,0.2,0.1,-0.3,0),S0=1,r=0,q=0,T=5,k=1)
```

```
## [1] 0.2624349
```

(b)

```r
library(nloptr)
```

```
## Warning: package 'nloptr' was built under R version 3.3.3
```

```r
setwd("E:/621 computational method/H5")
data <- read.csv("AMZN170505.csv")

square_error <- function(params){
  kappa = params[1];
  theta = params[2];
  sigma = params[3];
  V0    = params[4];
  rho   = params[5];
  lambda = params[6];

  #known parameters
  callprice.mkt <- (data$Bid+data$Ask)/2
  strike.mkt <- data$Strike
  S0 <- 909.28 #close price of AMZN at April 05,2017
  implied.vol <- data$Implied.Volatility
  T <- 30/252

  return(sum(callprice.mkt-Heston_Price(params,S0,r=0.005,q=0,T,strike.mkt))^2)
}

#Optimization with Inequality Constraints
g.ineq <- function(params){
  return(list( "constraints"= 2*params[1]*params[2]-params[3]^2,"jacobian"= 1))
}

x0=c(0.01,0.01,0.01,0.01,0.01,0.01);ub=Inf;lb=-Inf
local_opts <- list( "algorithm" = "NLOPT_LD_MMA", "xtol_rel"  = 1.0e-7 )
opts <- list( "algorithm" = "NLOPT_LD_AUGLAG",
              "xtol_rel"  = 1.0e-7,
              "maxeval"   = 1000,
              "local_opts" = local_opts )
#res2a <- nloptr( x0=x0,eval_f=square_error,lb=lb,ub=ub, eval_g_ineq=g.ineq,opts=opts)
#print( res2a )
```

(c)

```r
#Local (deterministic) algorithms

#Stochastic algorithms
```