MATLAB File Descriptions and Codes

**knock\_in\_put.m**

(MATLAB function to price a knock in put using Monte Carlo Simulations)

function[V] = knock\_in\_put(r,SDx,x0,T,n,N,K,Barrier)

dt = T/n;

for i = 1:N

x = zeros(1,n);

x(1) = x0;

for j = 1:n

x(j+1) = x(j) + x(j)\*r\*dt + x(j)\*SDx\*sqrt(dt)\*randn();

end

if min(x) <= Barrier

payoff(i) = max(K - x(n), 0) \* exp(-r\*T);

else

payoff(i) = 0;

end

V = mean(payoff);

end

**call\_bsm.m**

(MATLAB function to price a call option using Black Scholes Model)

function [Value\_Call] = call\_bsm(sigma,T,S0,r,K)

d1 = (log(S0/K)+(r+(sigma^(2)/2))\*T)/(sigma\*sqrt(T));

d2 = d1 - sigma\*sqrt(T);

Value\_Call = S0\*normcdf(d1) - K\*exp(-r\*T)\*normcdf(d2);

end

**HestonSV\_room.m**

(MATLAB file to reproduce Figure 4. Calls another function mentioned below)

clear all

clc

kappa\_star = 2; %speed of mean reversion

theta\_star = 0.01/252; %daily long term variance of v

v0 = 0.01/252; %daily current variance of underlying

rho = 0; %correlation between dz1 and dz2

T = 0.5\*252; %time to maturity in days

r = 0; %daily rate of interest

K = 100; %strike price of call

S0 = 100; %current underlying price

sigma = 0.1/sqrt(252); %daily volatility of v

S = 70:130; %range of spot prices for the graph

V\_HSV = zeros(length(S),1); %vector to store Heston Model Prices

V\_BS = zeros(length(S),1); %vector to store Black Scholes Prices

for i = 1:length(S)

V\_HSV(i) = StochVol(S(i),K,sigma,T,r,v0,rho,kappa\_star,theta\_star);

V\_BS(i) = call\_bsm(sqrt(v0),T,S(i),r,K);

end

diff1 = V\_HSV - V\_BS; %difference between prices

sigma = 0.2/sqrt(252); %daily volatility of v

S = 70:130; %range of spot prices for the graph

V\_HSV = zeros(length(S),1); %vector to store Heston Model Prices

V\_BS = zeros(length(S),1); %vector to store Black Scholes Prices

for i = 1:length(S)

V\_HSV(i) = StochVol(S(i),K,sigma,T,r,v0,rho,kappa\_star,theta\_star);

V\_BS(i) = call\_bsm(sqrt(v0),T,S(i),r,K);

end

diff2 = V\_HSV - V\_BS; %difference between prices

plot(S,diff1,'r',S,diff2,'b--'); %plot the Figure 4 in Heston(1993)

xlabel('Spot Price')

ylabel('Call Price Difference')

title('Difference between Heston Stochastic Vol Model and Black Scholes')

legend('sigma = 0.1','sigma = 0.2')

**StochVol.m**

(MATLAB function. Description given below)

function Call\_Price = StochVol(S0,K,sigma,T,r,v0,rho,kappa\_star,theta\_star)

%The function calulates the price of a call using the stochastic

%volatility model used by Henston in his 1993 paper

Call\_Price = S0\*(0.5+(1/pi)\*integral(@function\_in\_integral\_1,eps,100))-K\*exp(-r\*T)\*(0.5+(1/pi)\*integral(@function\_in\_integral\_2,eps,100));

function z1 = function\_in\_integral\_1(phi)

z1 = real((K.^(-i\*phi)).\*f1(i\*phi)./(i\*phi));

end

function z2 = function\_in\_integral\_2(phi)

z2 = real((K.^(-i\*phi)).\*f2(i\*phi)./(i\*phi));

end

function f1 = f1(phi)

a = kappa\_star\*theta\_star; %since it is equal to kappa\*theta

b1 = kappa\_star - rho\*sigma; %since kappa + lambda = kappa\_star

b2 = kappa\_star; %since kappa + lambda = kappa\_star

u1 = 0.5;

u2 = -0.5;

x = log(S0);

phi = phi';

d1 = sqrt(((rho.\*sigma.\*phi - b1).^2) - (sigma.^2)\*(2.\*u1.\*phi + (phi).^2));

g1 = (b1 - rho.\*sigma.\*phi + d1)./(b1 - rho.\*sigma.\*phi - d1);

C1 = r.\*phi.\*T + (a./sigma.^(2)).\*((b1 - rho.\*sigma.\*phi + d1).\*T - 2.\*log((1 - g1.\*exp(d1.\*T))./(1 - g1)));

D1 = ((b1 - rho.\*sigma.\*phi + d1)/(sigma.^(2))).\*((1 - exp(d1.\*T))./(1 - g1.\*exp(d1.\*T)));

f1 = exp(C1 + D1.\*v0 + phi.\*x);

f1 = f1';

end

function f2 = f2(phi)

a = kappa\_star\*theta\_star; %since it is equal to kappa\*theta

b1 = kappa\_star - rho\*sigma; %since kappa + lambda = kappa\_star

b2 = kappa\_star; %since kappa + lambda = kappa\_star

u1 = 0.5;

u2 = -0.5;

x = log(S0);

phi = phi';

d2 = sqrt(((rho.\*sigma.\*phi - b2).^2) - (sigma.^2)\*(2.\*u2.\*phi + (phi).^2));

g2 = (b2 - rho.\*sigma.\*phi + d2)./(b2 - rho.\*sigma.\*phi - d2);

C2 = r.\*phi.\*T + (a./sigma.^(2)).\*((b2 - rho.\*sigma.\*phi + d2).\*T - 2.\*log((1 - g2.\*exp(d2.\*T))./(1 - g2)));

D2 = ((b2 - rho.\*sigma.\*phi + d2)/(sigma.^(2))).\*((1 - exp(d2.\*T))./(1 - g2.\*exp(d2.\*T)));

f2 = exp(C2 + D2.\*v0 + phi.\*x);

f2 = f2';

end

end

**HestonMC\_root.m**

(MATLAB file to call Monte Carlo function and compute difference)

kappa\_star = 2; %speed of mean reversion

theta\_star = 0.01/252; %daily long term variance of v

v0 = 0.01/252; %daily current variance of underlying

rho = 0; %correlation between dz1 and dz2

sigma1 = 0.1/sqrt(252); %daily volatility of v

sigma2 = 0.2/sqrt(252); %daily volatility of v

T = 0.5\*252; %time to maturity in days

r = 0; %daily rate of interest

K = 100; %strike price of call

S0 = 100; %current underlying price

n = T; %discretizations of time

N = 100000; %number of simulations

valueMC\_1 = Heston1993\_MC(kappa\_star,theta\_star,v0,rho,sigma1,T,r,K,S0,n,N)

valueMC\_2 = Heston1993\_MC(kappa\_star,theta\_star,v0,rho,sigma2,T,r,K,S0,n,N)

valueF\_1 = StochVol(S0,K,sigma1,T,r,v0,rho,kappa\_star,theta\_star);

valueF\_2 = StochVol(S0,K,sigma2,T,r,v0,rho,kappa\_star,theta\_star);

diff\_1 = abs(valueMC\_1-valueF\_1)

diff\_2 = abs(valueMC\_2-valueF\_2)

**Heston1993\_MC.m**

(MATLAB function that calculates call price using Monte Carlo Simulations)

function value\_of\_call = Heston1993\_MC(kappa\_star,theta\_star,v0,rho,sigma,T,r,K,S0,n,N)

dt = T/n; %time step

payoff = zeros(N,1); %vector to store payoffs for each simulation

for j = 1:N

S = zeros(n+1,1);

v = zeros(n+1,1);

S(1) = S0;

v(1) = v0;

for i=1:n

%generate rho-correalted variables

eta1 = randn();

eta2 = eta1\*rho + randn()\*sqrt(1 - rho^2);

%recursive process to get values of S(i+1) and v(i+1) from

%S(i) and v(i)

S(i+1) = S(i)\*exp((r - 0.5\*max(v(i),0))\*dt ...

+ sqrt(max(v(i),0))\*sqrt(dt)\*eta1);

v(i+1) = v(i) + kappa\_star\*(theta\_star - max(v(i),0))\*dt ...

+sigma\*sqrt(max(v(i),0))\*sqrt(dt)\*eta2;

end

payoff(j) = exp(-r\*T)\*max(S(n+1) - K,0); %discounted payoff

end

value\_of\_call = mean(payoff); %average over all simulations

end

**NandiGARCH\_root.m**

(MATLAB file that calls the Nandi GARCH formula function)

lambda\_star = -0.5; %The risk neutral lamda (risk premiuim)

S0 = 100; %The spot price

K = 100; %The strike price

h0 = (0.15\*0.15)/252; %The inital conditional variance

T1 = 50; %The time to maturiy for the first option

T2 = 100; %The time to maturity for the second option

alpha1 = 1.32\*10^(-6); %The kurtosis (speed of mean reversion)

beta1 = 0.589; %The annual long term variance of h

gamma1 = 421.39; %The skewness of the distribution

omega = 5.02\*10^(-6); %The intercept of the conditional varaince

lambda = 0.205; %The risk premium

r = 0; %The risk free rate

NandiGARCH(S0,K,h0,T1,r,lambda\_star,alpha1,beta1,gamma1,lambda,omega)

NandiGARCH(S0,K,h0,T2,r,lambda\_star,alpha1,beta1,gamma1,lambda,omega)

**NandiGARCH.m**

(MATLAB function. Description given below)

function Price=NandiGARCH(S0,K,sigma,T,r,lambda\_star,alpha1,beta1,gamma1,lambda,omega)

%The function calulates the price of a call using the changing

%volatility model used by Henston and NANDI in their 2000 paper

Price = S0\*0.5+((1/pi)\*integral(@Integrand1,0,Inf))...

-K\*exp(-r\*T)\*(0.5+(1/pi)\*integral(@Integrand2,0,Inf));

function z1=Integrand1(phi)

%function that returns the function to be integrated

z1 = real((K.^(-i\*phi).\*momgfun(i\*phi+1))./(i\*phi));

end

function z2=Integrand2(phi)

%function that returns the function to be integrated

z2 = real((K.^(-i\*phi).\*momgfun(i\*phi))./(i\*phi));

end

function f1=momgfun(phi)

% function that returns the value for the moment generating function

gamma\_star = gamma1 + lambda + 0.5;

phi = phi';

A(:,T) = phi\*r; %to get vector of zeros since r = 0 at time = T

B(:,T) = phi\*r; %to get vector of zeros since r = 0 at time = T

for i=1:T-1

A(:,T-i) = A(:,T-i+1)+phi.\*r+B(:,T-i+1).\*omega...

-0.5\*log(1-2\*alpha1.\*B(:,T-i+1));

B(:,T-i) = phi.\*(lambda\_star+gamma\_star)-.5\*gamma\_star^2+beta1.\*B(:,T-i+1)...

+0.5.\*(phi-gamma\_star).^2./(1-2.\*alpha1.\*B(:,T-i+1));

end

A\_t0 = A(:,1)+phi.\*r+B(:,1).\*omega...

-0.5\*log(1-2.\*alpha1.\*B(:,1));

B\_t0 = phi.\*(lambda\_star+gamma\_star)-.5\*gamma\_star^2+beta1.\*B(:,1)...

+0.5\*(phi-gamma\_star).^2./(1-2.\*alpha1.\*B(:,1));

f1 = S0.^phi.\*exp(A\_t0+B\_t0.\*sigma);

f1 = f1';

end

end

**NandiGARCH\_MC\_root.m**

(MATLAB file that calls the GARCH Monte Carlo Simulation and computes difference)

lambda\_star = -0.5; %The risk neutral lamda (risk premiuim)

S0 = 100; %The spot price

K = 100; %The strike price

h0 = (0.15\*0.15)/252; %The inital conditional variance

T1 = 50; %The time to maturiy for the first option

T2 = 100; %The time to maturity for the second option

n = 10000; %The number of time steps

N = 100000; %The number of simulations

alpha1 = 1.32\*10^(-6); %The kurtosis (speed of mean reversion)

beta1 = 0.589; %The annual long term variance of h

gamma1 = 421.39; %The skewness of the distribution

omega = 5.02\*10^(-6); %The intercept of the conditional varaince

lambda = 0.205; %The risk premium

r = 0; %The risk free rate

p\_MC(1) = GARCH2000\_MC(T1,r,K,S0,h0,T1,n,lambda,lambda\_star,alpha1,beta1,gamma1,omega);

p\_MC(2) = GARCH2000\_MC(T2,r,K,S0,h0,T2,n,lambda,lambda\_star,alpha1,beta1,gamma1,omega)

p(1) = NandiGARCH(S0,K,h0,T1,r,lambda\_star,alpha1,beta1,gamma1,lambda,omega);

p(2) = NandiGARCH(S0,K,h0,T2,r,lambda\_star,alpha1,beta1,gamma1,lambda,omega);

diff\_bw\_MC\_Formula = max(abs(p\_MC-p))

**GARCH2000\_MC.m**

(MATLAB function that calculates call price using Monte Carlo Simulation of GARCH Model)

function value\_of\_call = GARCH2000\_MC(T,r,K,S0,h0,n,N,lambda,lambda\_star,alpha1,beta1,gamma1,omega)

gamma1\_star = gamma1 + lambda + 0.5;

dt = T/n; %time step

payoff = zeros(N,1); %vector to store payoffs for each simulation

for j = 1:N

S = zeros(n+1,1);

h = zeros(n+1,1);

z\_star = zeros(n+1,1);

S(1) = S0;

h(1) = h0;

z\_star(1) = randn()\*sqrt(dt);

for i=1:n

%recursive process to get values of S(i+1) and h(i+1) from

%S(i) and h(i)

h(i+1) = omega + beta1\*h(i) + alpha1\*((z\_star(i) ...

- gamma1\_star\*sqrt(max(h(i),0))))^(2);

%generate Weiner Process

z\_star(i+1) = randn()\*sqrt(dt) + (lambda + 0.5)\*sqrt(h(i+1));

S(i+1) = S(i)\*exp(((r + lambda\_star\*h(i+1))\*dt ...

+ sqrt(max(h(i+1),0))\*z\_star(i+1)));

end

payoff(j) = exp(-r\*T)\*max(S(n+1) - K,0); %discounted payoff

end

value\_of\_call = mean(payoff); %average over all simulations

end