%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% FRE 6251 Numerical and Simulation Techniques in Finance

% Assignment #3

% Name: Surya L Gurung ID: 0449604

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Brief Reports of Assignment #3:

(1). This simulation has two matlab files, one is a function called “ MertonOptionPrice(S, K, r, sigma, lambda, T, cORp, n, path) ”, and other one is matlab script file called “MertonOptionMCS.m”.

The function MertonOptionPrice() uses Monte Carlo Simulation method to calculate European Option Price with jump-diffusion ,model and estimated error of the payoff values of n simulations.

I used example of an asset with following parameters and values:

S = initial price of asset

K = strike price

r = risk-free rate

sigma = standard deviation of asset

lambda = expected jump number per year

T = time to maturity in year

cORp = option type, 1 for call and -1 for put put option

n = number of time steps

path = number of paths

For example,

S = 100;

K = 115;

r = 0.05;

sigma2= 0.20;

lambda = 4;

T = 1;

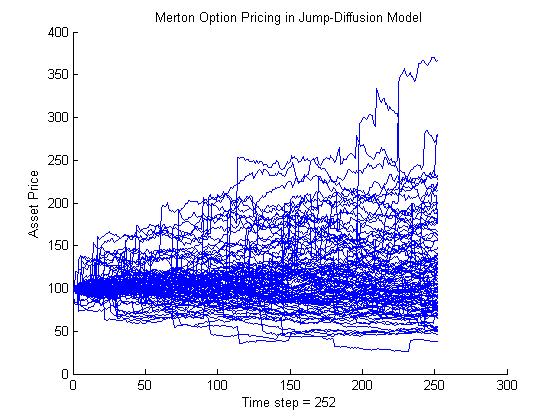
cORp = 1;

time step = 20;

paths = 10;

This assets example is used in MertonOptionMCS.m file using MertonOptionPrice.m function. This is executed initially for 10 paths with 20 time steps simulations and then for up to 1,000,000 paths in multiple of 10 in each iteration.

The results of option price and error estimation is shown for each group of simulation in the following page. For a review, a graph of jump-diffusion of 100 simulations path with 252 time steps is also included.



**The result of running the “MertonOptionMCS.m” for upto 100,000 paths.**

>> MertonOptionMCS

path Option Price Error Estimation

------ --------------- -------------------

10 6.77648878796102 13.2819180244036

100 26.4361698166991 12.0363159362368

1000 20.0488529929224 2.60453506962599

10000 22.1021638039713 0.882428068400574

100000 22.2503488679784 0.28245526350862

>>

%%% File “MertonOptionMCS.m”

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% FRE 6251 Numerical and Simulation Techniques in Finance

% Assignment #3

% Name: Surya L Gurung ID: 0449604

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Monte Carlo Simulation of Jump-Diffusion Mertion Option

%-----------------------------------------------------------

% Parameter Description:

% S = initial price of asset

% K = strike price

% r = risk-free rate

% sigma = standard deviation of asset

% lambda = expected jump number per year

% T = time to maturity in year

% cORp = option type, 1 for call and -1 for put put option

% n = number of time steps

% path = number of paths

%----------

% Example

%----------

S = 100;

K = 115;

r = 0.05;

sigma = 0.20;

lambda = 4;

T = 1;

cORp = 1; % this is for call option

n = 20;

path = 100;

row = 1;

format long g

disp([' path', ' Option Price', ' Error Estimation'])

disp([' ------', ' ---------------', ' -----------------'])

while path < 100000

[optionPrice, error] = MertonOptionPrice(S, K, r, sigma, lambda, T, cORp, n, path);

result(row,:) = [path, optionPrice, error];

path = path \* 10;

row = row + 1;

end

disp(result);

% Function “MertonOptionPrice.m”

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% FRE 6251 Numerical and Simulation Techniques in Finance

% Assignment #3

% Name: Surya L Gurung ID: 0449604

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [optionPrice, error]=MertonOptionPrice(S, K, r, sigma, lambda, T, cORp, n, path)

% S = initial price of asset

% K = strike price

% r = risk-free rate

% sigma = standard deviation of asset

% lambda = expected jump number per year

% T = time to maturity in year

% cORp = option type, 1 for call and -1 for put put option

% n = number of time steps

% path = number of paths

dT = T/n;

%Z = randn(1,n); % random number vector of 1xn size drawn from standard normal distribution.

%N = poissrnd(lambda \* dT , 1, n); % 1xn matrix of random variable drawn from poisson distribution

Xt(1) = log(S); % log of initial asset price

format long g

a = - 5;

b = .5;

m = exp(a + 0.5 \* b^2)

mu = r - lambda \* m

hold on

for i = 1:path

for j = 1:n

Z = normrnd(0,1);

N = poissrnd(lambda \* dT);

if N == 0 % No jump

M = 0;

else % There is jump

%j = 9

Yj = lognrnd(mu, sigma, 1, N); % Generating Yi

M = sum(log(Yj)) ; % Sum of Log(Yi)

end

Xt(j+1) = Xt(j) + (mu - 0.5\*sigma^2) \* dT + sigma \* sqrt(dT) \* Z + M;

end

if (path < 10000)

plot(0:n, exp(Xt));

title('Merton Option Pricing in Jump-Diffusion Model');

xlabel(['Time step = ', num2str(n)]);

ylabel('Asset Price');

end

optionVals(i) = exp(-r\*T)\* max(cORp\*(exp(Xt(n+1)) - K),0);

end

hold off

optionPrice = mean(optionVals);

error = 1.96 \* (std(optionVals)/sqrt(path));