Application of Matlab for Finance
Class 6

Shi, Yining

October 14, 2017

Today's Class

- Simulation
- Normal Stock Price Model
- Log Normal Stock Price Model
- Black-Scholes Option Pricing

Random Number Generators

- `rand(m,n)`: uniform random number on the interval (0,1)

- `randi(m,n)`: uniform integer random number on the interval (0,1)
- `randn(m,n)`: standard Normal random number

- `normrnd(mu,sigma,m,n)`: normal random number with mean `mu` and standard deviation `sigma`
- `trnd(nu,m,n)`: student t-distribution random number with `nu` degrees of freedom

- `randg(m,n)`: standard Gamma random number
- `(m,n)` defined the output matrix size, `m`-by-`n`, that stores the simulated numbers.

Examples: Normal & Student-t

- Simulate 100×1 standard Normal random variables
- Simulate 100×1 Normal random variables from $\mathcal{N}(0.08, 0.2^2)$ use both `randn` and `normrnd`

$$X \sim \mathcal{N}(\mu, \sigma^2)$$
$$X = \mu + \sigma * \mathcal{N}(0, 1)$$

- Simulate 100×1 random variables from a Student t with 8 degrees of freedom

```
% Simulate 100 number from N(0,1)
x = randn(100,1);

% Simulate 100 number from N(0.08, 4)
% X = mu + sigma* N(0,1)
y = 0.08 + 0.2 * randn(100,1);
y2 = normrnd(0.08,0.2,[100,1]);

% Simulate student t-distribution
x = trnd(8,100,1);
```

Set Seeds for Random Generator

- Sometimes, we want to use the same sequence of random numbers for various reasons, such as code debugging or to generate reproductive results.

- We can set the seed for the random number generator using `rng`.
- Call the stored seed every time when we want to regenerate the same sequence of random numbers.


```
% seeding the random number generator
s = rng; % set the seed for generator
a = rand(1,5)

rng(s); % call the stored seed
b = rand(1,5)
```

Simulate Asset Prices

- In finance, the price of a particular stock at a future time t is unknown at the present

- We often think of it as being a random variable S_t
- If we simplly assume S_t follows a normal distribution $\mathcal{N}(\mu, \sigma)$ for $t = 1, 2, 3, \ldots$, then

$$S_1 = \mu + \sigma * \epsilon_1$$

$$S_2 = \mu + \sigma * \epsilon_2$$

$$\Delta S = S_2 - S_1 = 0 + \sigma * (\epsilon_2 - \epsilon_1)$$

- In expectation, $E(\Delta S) = E[\sigma * (\epsilon_2 - \epsilon_1)] = 0$ since both ϵ_1 and ϵ_2 are random draws from $\mathcal{N}(0, 1)$.
- We need the time dimension variations.

Normal Stock Price Model

- We assume the stock price follows a stochastic process

$$\Delta S = S_0(\mu\Delta t + \sigma\sqrt{\Delta t}\epsilon)$$

$$\epsilon \sim \mathcal{N}(0,1) \quad \Delta t = t - 0 = t$$

- Then $S_t = S_0 + \Delta S = S_0(1 + \mu\Delta t + \sigma\sqrt{\Delta t}\epsilon)$
- This is process known as the Brownian Motion
- Then gross return and net return on stock as

$$R_t = \frac{S_t}{S_0} = 1 + \mu\Delta t + \sigma\sqrt{\Delta t}\epsilon$$

$$R_t^{net} = R_t - 1 = \mu\Delta t + \sigma\sqrt{\Delta t}\epsilon$$

Simulate the Stock Price Process

- Consider stock with annual return of 0.15 and annual volatility of 0.3, today you observe its price $1. Simulate a 1-year path for this stock price with $\Delta t = \frac{1}{250}$ year.


```
dt = 1/250;
T = 1;
tgrid = 0: dt : T;
N = length(tgrid);

% Set up parameters & initialize the price vector
S0 = 1;
mu = 0.15;
sigma = 0.3;
S = zeros(1,N);

% Simulate random number epsilon
eps = randn(1,N);
% Simulate stock prices
S = S0*(1 + mu*tgrid + sigma*sqrt(tgrid).*eps);
plot(tgrid, S)
legend( 'S')
xlabel('Time(yr)')
ylabel('Asset Price($)')
```

Simulate the Stock Price Process: 3 Stocks

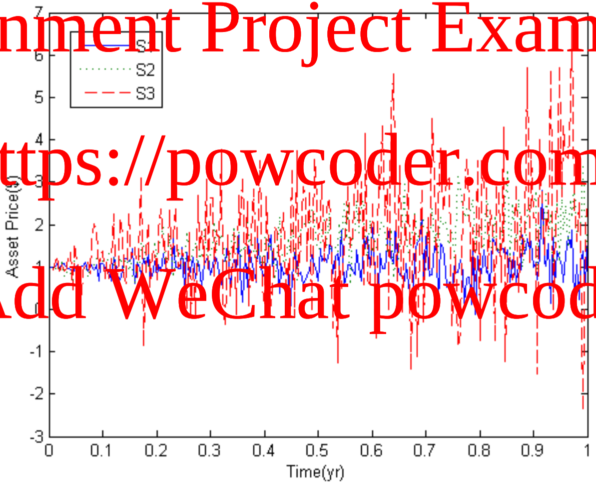

```
% Define mean & volatility
mu1 = 0.05; sigma1 = 0.6;
mu2 = 1.2;  sigma2 = 0.6;
mu3 = 1.2;  sigma3 = 2;

% dt = 1/250 yr, T = 1 yr,
% time gride = T/dt + 1 = 251 grid points
dt = 1/250;
T = 1;
tgrid = 0: dt: 1;
N = length(tgrid);
S1 = zeros(1,N); S2 = zeros(1,N); S3 = zeros(1,N);
S0 = 1;

% Simulate 3 random numbers epsilon
eps = randn(3,N);

% Simulate stock prices
S1 = S0 *(1+ mu1*tgrid + sigma1*sqrt(tgrid).* eps(1,:));
S2 = S0 *(1+ mu2*tgrid + sigma2*sqrt(tgrid).* eps(2,:));
S3 = S0 *(1+ mu3*tgrid + sigma3*sqrt(tgrid).* eps(3,:));
plot(tgrid, S1, '-', tgrid, S2,':', tgrid, S3,'--')
legend( 'S1', 'S2', 'S3')
xlabel('Time(yr)')
ylabel('Asset Price($)')
```

Normal Stock Price Model



Log-Normal Stock Price Model

- Issues with Normal stock price: negative stock prices
- $\Delta \ln(S) = \ln S_t - \ln S_0 = (\mu - \frac{\sigma^2}{2})\Delta t + \sigma\sqrt{\Delta t}\epsilon$ wherein $\epsilon \sim N(0,1)$, and $\Delta t = t - 0 = t$
- S_t follows a log-normal distribution, as often referred as Geometric Brownian Motion:

$$S_t = S_0 e^{(\mu - \frac{\sigma^2}{2})\Delta t + \sigma\sqrt{\Delta t}\epsilon}$$

$$\ln(S_t) = \ln(S_0) + (\mu - \frac{\sigma^2}{2})\Delta t + \sigma\sqrt{\Delta t}\epsilon$$

$$r_t = \ln(S_t) - \ln(S_0) = \ln(\frac{S_t}{S_0})$$

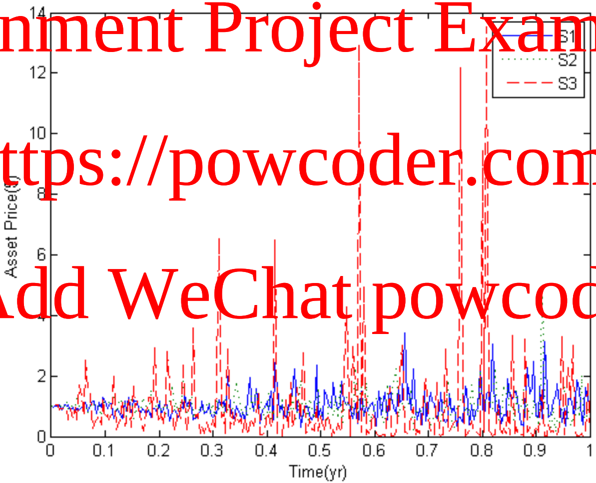
$$= (\mu - \frac{\sigma^2}{2})\Delta t + \sigma\sqrt{\Delta t}\epsilon$$

Log Normal Stock Price Model


```
mu1 = 0.05; sigma1 = 0.6;
mu2 = 1.2;  sigma2 = 0.6;
mu3 = 1.2;  sigma3 = 2;

% dt = 1/250 yr, T = 1 yr,
% time gride = T/dt +1 = 251 grid points
dt = 1/250
T = 1;
tgrid = 0: dt: T;
N = length(tgrid);
S1 = zeros(1,N); S2 = zeros(1,N); S3 = zeros(1,N);
S0 = 1;
% Simulate random number epsilon
eps = randn(3,N);
S1=S0*exp((mu1-0.5*(sigma1^2))*tgrid+sigma1*sqrt(tgrid).*eps);
S2=S0*exp((mu2-0.5*(sigma2^2))*tgrid+sigma2*sqrt(tgrid).*eps);
S3=S0*exp((mu3-0.5*(sigma3^2))*tgrid+sigma3*sqrt(tgrid).*eps);
plot(tgrid, S1, '-', tgrid, S2,':', tgrid, S3,'--')
legend( 'S1', 'S2', 'S3')
xlabel('Time(yr)')
ylabel('Asset Price($)')
```

Log-Normal Stock Price Model



Options Pricing

- $V(S,t)$ is the value of an option
  - $C(S,t)$: *Call* options give the right to purchase the underlying asset at future expiry date with agreed price today.
  - $P(S,t)$: *Put* options give the right to sell the underlying asset at future expiry date with agreed price today.
- S the value of stock price (i.e. underlying asset)
- K the strike price of the option contract (i.e. the agreed price)
- T is the maturity of the contract (i.e. the future expiry date)
- r is the risk free rate
- σ is the volatility of the underlying stock.

With above notation, the payoffs of European Calls and Puts at the expire date is:

$$C(S,T) = \max(S-K,0) \qquad P(S,T) = \max(K-S,0)$$

Option Pricing Simulation

- Use the normal stock price model simulate $10,000,000$ scenarios for a stock that with $S_0 = 100, T = 1, \mu = 10\%, \sigma = 20\%$;
- Calculate the expected price of an European Call and Put option on the this stock with $K = 100, r = 5\%$
  - Today's price is the discounted value of expected future payoffs
  - Law of Large Number implies the sample mean converts to the true mean of the underlying distribution.
- Note: in the previous exercises, we simulate over a sequence of time grids in the future.
- Note: this exercise is different as we simulate at only one time point in the future (ie, the maturity date T), but with different $10,000,000$ scenarios.

Option Pricing Exercises 1


```
S0 = 100; % Value of the underlying
K = 100; % Strike (exercise price)
T = 1;    % Maturity
mu = 0.10; % Stock price mean
r = 0.10; % Risk free interest rate
sigma = 0.20; % Volatility

M=10000000; % Number of simulation trials
eps = randn(M,1); %
S_T=S0*exp((mu-0.5*(sigma^2))*T+sigma*sqrt(T).*eps);

S_T = S0*(1 + mu*T+ sigma*sqrt(T).*eps);
payoff_call=max(S_T-K,0);
payoff_put=max(K-S_T,0);
p_call = mean(exp(-r*T)*payoff_call); % continuous discount
p_put = mean(exp(-r*T)*payoff_put);
```

Options Price 2: The Black-Scholes Formula

- The price of a call option is given by

$$C(T) = SN(d_1) - Ke^{-rT}N(d_2)$$

- The price of a put option is given by

$$P(T) = Ke^{-rT}N(-d_2) - SN(-d_1)$$

- where

$$d_1 = \frac{\ln(S/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}$$

$$d_2 = d_1 - \sigma\sqrt{T}$$

- and $N(d_1)$ and $N(d_2)$ denotes the standard cumulative normal probability for the values of d_1 and d_2. It is the probability that a random draw from a normal distribution.

Option Pricing Exercises

- Exercise 2:
  - Create a function perform the Black-Scholes Formula to determine the above options' price
  - The function shall allows the user to define a option as call or put
  - Compare with the above method
    - Currently, the normal stock price process is used in the simulation based method;
    - Black-Scholes model assumes a log-normal stock price;
    - The difference among the two comes from the different underlying stock price processes;

Option Pricing Exercises 2: Function


```
function price = BlackScholesPrice(S, K, T, r, sigma, CallorPut)
% This function calculates option price base on the Black—Schole formula.

% Input: S: spot stock price
%        K: strike price
%        T: maturity
%        r: interest rate
%        sigma: volatility
%        callorput: user—defined string input as 'Call' or 'Put' option

if strcmp(CallorPut,'Call') == 1
    phi = 1;
elseif strcmp(CallorPut,'Put') == 1
    phi = —1;
else
    error('Invalid Option Type')
end

d1 = (log(S/K) + (r + 0.5 * sigma^2)* T)./ sigma.* sqrt(T);
Nd1 = normcdf(phi*d1,0,1);

d2 = d1 — sigma.* sqrt(T);
Nd2 = normcdf(phi*d2,0,1);

price = phi.*S.*Nd1 — phi.*K.*exp(—r * T).*Nd2;
end
```

Option Pricing Exercises 2: Main Command


```
%%%%%%%%%% Option parameters %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
S = 100; % Value of the underlying
K = 100; % Strike (exercise price)
T = 1; % Maturity
r = 0.10; % Risk free interest rate
sigma = 0.20; % Volatility
% parameters = [S, K, T, r, sigma];
% call functions to calculate the price of the option
CallPrice = BlackScholesPrice(S, K, T, r, sigma, 'Call');
PutPrice = BlackScholesPrice(S, K, T, r, sigma, 'Put');
% fprintf output
fprintf('The Price of a European Call is : %.2f\n', CallPrice);
```

Take Away

- Basic Simulation Code
- Simulate stock prices follow normal and log normal process
- Pricing Options using simulation and Black-Scholes Model
- In your coursework, you will use the log normal stock price for simulation. Will there be any difference between the option price based on the simulation versus the Black-Scholes?