Application of Matlab for Finance

October 14, 2017

Today's Class

-
-
-
- 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)
-
-
- `trnd(nu,m,n)`: student t-distribution ran
  degrees of freedom
- `randg(m,n)`: standard Gamma random nu
- `(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 $N(0.08, 0.2^2)$ use both `randn` and `normrnd`
- $\times$ grees 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.
-
  rng.
- same sequence of random numbers.


```
1  % seeding the random number generator
2  s = rng; % set the seed for generator
3  a = rand(1,5)
4
5  rng(s); % call the stored seed
6  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.
-
- I) for

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

- 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

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

$$\epsilon \qquad (0,1) \quad \Delta t = t \quad 0 = t$$

-
-
-

- Then gross return and net return on stock as

$$R_t = \frac{S_t}{S_0} = 1 + \mu$$

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

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.


```
d
T = 1
t
N = 1

% 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


```
1   % Define mean & volatility
2   mu1 = 0.05; sigma1 = 0.6;
3   mu2 = 1.2;  sigma2 = 0.6;
4   mu3 = 1.2;  sigma3 = 2;
5
6   % d
7   % t
8   d
9   T = 1
10  t
11  N = 1
12  S
13  S0 = 1;
14
15  % Simulate 3 random numbers epsilon
16  eps = randn(3,N);
17
18  % Simulate stock prices
19  S1 = S0 *(1+ mu1*tgrid + sigma1*sqrt(tgrid).* eps(1,:
20  S2 = S0 *(1+ mu2*tgrid + sigma2*sqrt(tgrid).* eps(2,:
21  S3 = S0 *(1+ mu3*tgrid + sigma3*sqrt(tgrid).* eps(3,:
22  plot(tgrid, S1, '-', tgrid, S2,':', tgrid, S3,'--')
23  legend( 'S1', 'S2', 'S3')
24  xlabel('Time(yr)')
25  ylabel('Asset Price($)')
```

Normal Stock Price Model

Log-Normal Stock Price Model

- Issues with Normal stock price: negative stock prices
- $\Delta \ln(S) = \ln S_t - \ln S_0 = (\mu - \frac{\sigma^2}{2})\Delta t + \sigma\sqrt{\Delta t}\epsilon$ where $\epsilon \sim N(0,1)$, and $\Delta t = t - 0 = t$
-

$$\ln(S_t) = \ln(S_0) + (\mu - \quad$$

$$r_t = \ln(S_t) - \ln(S_0 \quad \_0$$

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

Log Normal Stock Price Model


```
mu1 = 0.05; sigma1 = 0.6;
mu2 = 1.2;  sigma2 = 0.6;
m

% d
% 
d
T = 1
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(tgri
S2=S0*exp((mu2-0.5*(sigma2^2))*tgrid+sigma2*sqrt(tgri
S3=S0*exp((mu3-0.5*(sigma3^2))*tgrid+sigma3*sqrt(tgri
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.
- T is the maturity of the contract (i.e. the future ex
- 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

- Note: in the previous exercises, we simulate over a 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 = 1
T = 1    % Ma
m
r = 0
s

M
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

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

-

-

$$d_1 = \frac{\ln(S/K) + (}{\sigma}$$

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

- 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

  - Currently, the normal stock price proces
    based method;
  - Black-Scholes model assumes a log-nor
  - The difference among the two comes from
    stock price processes;

Option Pricing Exercises 2: Function


```
1  function price = BlackScholesPrice(S, K, r, T, sigma, CallorPut)
2  % This function calculates option price base on the Black—Schole formula.
3
4  % Input: S: spot stock price
5  %        K: strike pr
6  %        T: maturit
7  %        r: interes
8  %        sigma: vol
9  %        CallorPut
10
11 i
12     phi = 1;
13 elseif strcmp(CallorPut,'Put') == 1
14     phi = —1;
15 else
16     error('Invalid Option Type')
17 end
18
19 d1 = (log(S/K) + (r + 0.5 * sigma^2)* T)./ sigma.* sq
20 Nd1 = normcdf(phi*d1,0,1);
21
22 d2 = d1 — sigma.* sqrt(T);
23 Nd2 = normcdf(phi*d2,0,1);
24
25 price = phi.*S.*Nd1 — phi.*K.*exp(—r * T).*Nd2;
26 end
```

Option Pricing Exercises 2: Main Command


```
%
S = 1
K = 1
T = 1 % M
r = 0
s
% p
% 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

-
-
-
- I
  simulation. Will there be any difference betwee
  based on the simulation versus the Black-Schol