Option Pricing Using Monte Carlo Method

Parameter for options and underlying asset

```
r = 0.02;       %risk-free rate
S0 = 100;       %current price of the underlying stock
sig = 0.25;     %volatility of stock
K = 105;        %strike price
steps = 8;      %number of step to compute (2 months x 4 week/month)
n_paths = 1000;  %number of paths
T = 8/52;       %time to maturity in years
dt = T/steps;   %simulation unit time in years
```

Simulating stock price paths

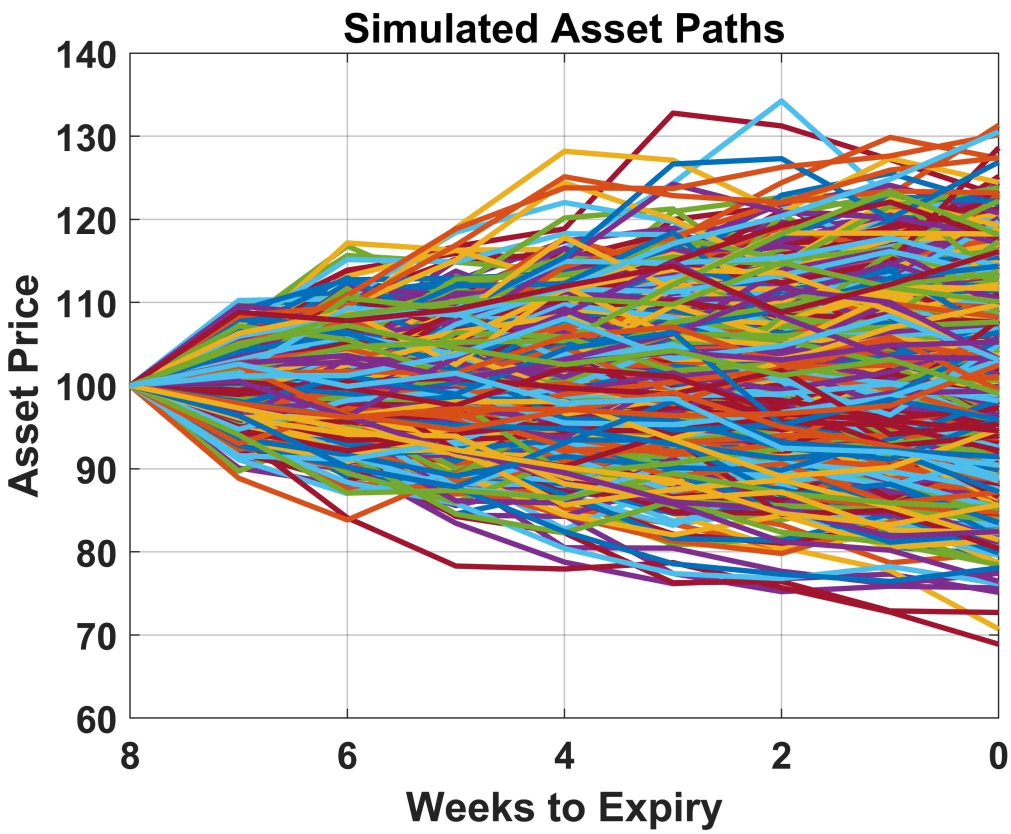
```
%fix the seed of random number generater for repeatability
rng(1);

%each column of S is a single path
%each row corrosponds to a single time step
S = S0*[ones(1,n_paths); ...
            cumprod(exp(r*dt - 0.5*sig*sig*dt +sig*sqrt(dt)*randn(steps,n_paths)),1)]
```

```
S = 9×1000
  100.0000  100.0000  100.0000  100.0000  100.0000  100.0000  100.0000  100.0000 ...
   97.7539   99.2983   93.3929   93.7182  101.5549  102.0917  103.2409  101.4654
  101.8180  101.3159   89.3505   90.3450  100.4051  100.1930   98.0503   97.0737
   99.1541   98.3462   93.0459   89.0278  103.1650   99.6342   96.1507   95.4338
   95.3916  101.0912   99.8056   93.4423  106.5177  101.7492   96.0351   97.3994
   92.6158   95.9165   98.8367   89.0956  104.6980   98.1099   91.2239   96.4495
   90.7755   98.8523  100.9051   88.1774  111.4048   96.9203   92.0264   90.4143
   89.0150   98.0025  107.3096   92.1294  113.6851   93.0430   85.7876   90.5225
   89.5478   98.5496  114.0905   92.3205  111.1826   90.8379   86.3650   91.3951
```

Visulize the paths

```
time = steps:-1:0;
plot(time,S,'Linewidth',2);
set(gca,'XDir','Reverse','FontWeight','bold','Fontsize',14);
xlabel('Weeks to Expiry','FontWeight','bold','Fontsize',14);
ylabel('Asset Price','FontWeight','bold','Fontsize',14);
title('Simulated Asset Paths','FontWeight','bold','Fontsize',24);
grid on
```




```
set(gcf,'Color','w');
```

Asian Call price

```
Asian_call_payoffs = (mean(S) - K).*((mean(S) - K) > 0);
Asian_call_price = exp(-r*T)*mean(Asian_call_payoffs)
```

```
Asian_call_price = 0.6768
```

```
stderr=std(exp(-r*T)*Asian_call_payoffs)/sqrt(n_paths);
CI_Asian_call_price = [Asian_call_price - 1.96*stderr, Asian_call_price + 1.96*stderr]
```

```
CI_Asian_call_price = 1×2
    0.5645    0.7891
```

Asian Put price

```
Asian_put_payoffs = (K - mean(S)).*((K - mean(S)) > 0);
Asian_put_price = exp(-r*T)*mean(Asian_put_payoffs)
```

```
Asian_put_price = 5.3789
```

```
stderr=std(exp(-r*T)*Asian_put_payoffs)/sqrt(n_paths);
CI_Asian_put_price = [Asian_put_price - 1.96*stderr, Asian_put_price + 1.96*stderr]
```

```
CI_Asian_put_price = 1×2
    5.0931    5.6647
```

Lookback Call price

```
lb_call_payoffs = (max(S) - K).*((max(S) - K) > 0);
lb_call_price = exp(-r*T)*mean(lb_call_payoffs)
```

```
lb_call_price = 3.2893
```

```
stderr=std(exp(-r*T)*lb_call_payoffs)/sqrt(n_paths);
CI_lb_call_price = [lb_call_price - 1.96*stderr, lb_call_price + 1.96*stderr]
```

```
CI_lb_call_price = 1×2
    2.9701    3.6085
```

Lookback Put price

```
lb_put_payoffs = (K - min(S)).*((K - min(S)) > 0);
lb_put_price = exp(-r*T)*mean(lb_put_payoffs)
```

```
lb_put_price = 10.7338
```

```
stderr=std(exp(-r*T)*lb_put_payoffs)/sqrt(n_paths);
CI_lb_put_price = [lb_put_price - 1.96*stderr, lb_put_price + 1.96*stderr]
```

```
CI_lb_put_price = 1×2
   10.3881   11.0795
```

Floating Lookback Call price

```
flb_call_payoffs = (S(end,:) - min(S)).*((S(end,:) - min(S)) > 0);
flb_call_price = exp(-r*T)*mean(flb_call_payoffs)
```

```
flb_call_price = 5.9949
```

```
stderr=std(exp(-r*T)*flb_call_payoffs)/sqrt(n_paths);
CI_flb_call_price = [flb_call_price - 1.96*stderr, flb_call_price + 1.96*stderr]
```

```
CI_flb_call_price = 1×2
    5.6077    6.3822
```

Floating Lookback Put price

```
flb_put_payoffs = (max(S) - S(end,:)).*((max(S) - S(end,:)) > 0);
flb_put_price = exp(-r*T)*mean(flb_put_payoffs)
```

```
flb_put_price = 6.2670
```

```
stderr=std(exp(-r*T)*flb_put_payoffs)/sqrt(n_paths);
CI_flb_put_price = [flb_put_price - 1.96*stderr, flb_put_price + 1.96*stderr]
```

```
CI_flb_put_price = 1×2
    5.9141    6.6200
```

American Put price

Least-Squares Monte Carlo (LSM) method proposed by Longstaff and Schwartz (2001)

```
%payoff matrix (payoff from exercising at each time step for each path)
h = (K-S).*((K-S) > 0)
```

```
h = 9×1000
    5.0000    5.0000    5.0000    5.0000    5.0000    5.0000    5.0000    5.0000 ···
    7.2461    5.7017   11.6071   11.2818    3.4451    2.9083    1.7591    3.5346
    3.1820    3.6841   15.6495   14.6550    4.5949    4.8070    6.9497    7.9263
    5.8459    6.6538   11.9541   15.9722    1.8350    5.3658    8.8493    9.5662
    9.6084    3.9088    5.1944   11.5577         0    3.2508    8.9649    7.6006
   12.3842    9.0835    6.1633   15.9044    0.3020    6.8901   13.7761    8.5505
   14.2245    6.1477    4.0949   16.8226         0    8.0797   12.9736   14.5857
   15.9850    6.9975         0   12.8706         0   11.9570   19.2124   14.4775
   15.4522    6.4504         0   12.6795         0   14.1621   18.6350   13.6049
```

```
% Value matrix (value/price of option at each time step for each path)
v = zeros(steps+1, n_paths);
%value of option at maturity (exercised at maturity)
v(end,:) = h(end,:)
```

```
v = 9×1000
         0         0         0         0         0         0         0         0 ···
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
   15.4522    6.4504         0   12.6795         0   14.1621   18.6350   13.6049
```

Compare immediate exercise value with continuation value obtained using condition expectation function (polynominal regression) and update value of option to payoff from exercising at given time step where it is optimal to exercise.

```
% repeat recusively upto time step 1
for j = steps:-1:2

    %Following steps until the next blank line are to find Ci
    %find paths where option is in-the-money at timestep j-1
    I = find(h(j,:));
    %obtain stock price at timestep j-1 where option is in-the-money
    X = S(j,I);
    %discount option value at timestep j to j-1 where option is in-the-money
    Y = v(j+1,I)*exp(-r*dt);
    %obtain condition expectation function (polynomial of degree 2)
    reg = polyfit(X, Y, 2);
    %find continuation value
    c = polyval(reg, X);

    %assign option value at timestep j-1
    v(j,:) = v(j+1,:)*exp(-r*dt);

    %find paths where it is optimal to exercise at timestep j-1
    II = nonzeros(I.*(h(j,I) > c))';
    %upadte the option values to payoffs from exercising at timestep j-1
    %where it is optimal
    v(j,II) =  h(j,II);
```

```
end
```

Discount option value for each path from time step 1 to time step 0.

```
v(1,:) = v(2,:)*exp(-r*dt);
v
```

```
v = 9×1000
   15.4047    6.4306   15.6374   14.6438         0   14.1185   18.5778   13.5631 ···
   15.4107    6.4330   15.6435   14.6494         0   14.1240   18.5849   13.5683
   15.4166    6.4355   15.6495   14.6550         0   14.1294   18.5920   13.5735
   15.4225    6.4380         0   12.6552         0   14.1348   18.5992   13.5787
   15.4284    6.4405         0   12.6601         0   14.1403   18.6064   13.5839
   15.4344    6.4429         0   12.6649         0   14.1457   18.6135   13.5892
   15.4403    6.4454         0   12.6698         0   14.1512   18.6207   13.5944
   15.4463    6.4479         0   12.6747         0   14.1566   18.6278   13.5996
   15.4522    6.4504         0   12.6795         0   14.1621   18.6350   13.6049
```

Take average of option values to find the price

```
%American put option price at time 0.
American_put_price = mean(v(1,:))
```

```
American_put_price = 6.9836
```

```
stderr=std(v(1,:))/sqrt(n_paths);
CI_American_put_price = [American_put_price - 1.96*stderr, American_put_price + 1.96*stderr]
```

```
CI_American_put_price = 1×2
    6.5458    7.4214
```