Q2

Reading Tips

Due to MATLAB settings, I need to put the simulation above the code for question. So, you will see my code in the 'Function' section at the bottom.

Evaluation results

```
% test
% define fixed constants
s0 = 100;
r = 0.06;
T = 1;
% Let's use the same M and N as Q1
M = 10000;
N = 1000;
% varying constants
Ks = [80, 90, 100, 110, 120];
sigmas = [0.01, 0.1, 0.3];
% option price matrix
op_mat = zeros(length(Ks), length(sigmas));
% standard error
er_mat = zeros(length(Ks), length(sigmas));
```

Turn off MATLAB warnings

```
w = warning('query','last');
id = w.identifier;
warning('off',id);
rmpath('folderthatisnotonpath');
```

```
Warning: "folderthatisnotonpath" not found in path.
```

Fill in the table

```
% MC
for i = 1:length(Ks)
    for j = 1:length(sigmas)
        K = Ks(i);
        sigma = sigmas(j);
        [op_price, stderr] = LSMC(s0, r, K, sigma, T, M, N);
        op_mat(i,j) = op_price;
        er_mat(i,j) = stderr;
    end
end
```

Next, add headers to the result

```
rowNames = {'K = 80', '90', '100', '110', '120'};
colNames = {'Sigma = 0.01', '0.1', '0.3'};
% option price table
op_tab = array2table(op_mat, "RowNames",rowNames, "VariableNames", colNames)
```

op_tab = 5×3 table

	Sigma = 0.01	0.1	0.3
1 K = 80	0	0	0.0041
2 90	0	0	0.1920
3 100	0.0582	0.8363	2.5455
4 110	9.4123	9.4141	9.7876
5 120	18.8295	18.8338	18.8933

```
% standard error table
er_tab = array2table(er_mat, "RowNames", rowNames, "VariableNames", colNames)
```

er_tab = 5×3 table

	Sigma = 0.01	0.1	0.3
1 K = 80	0	0	0.0001
2 90	0	0	0.0020
3 100	2.7396e-04	0.0028	0.0081
4 110	2.9868e-04	0.0030	0.0091
5 120	2.9494e-04	0.0029	0.0089

I used M = 10000 and N = 1000.

Function

The LSMC function estimates the time-0 price using the Least Square Monte Carlo method.

Input:

- s0: underlying asset price at time-0
- r: risk-free interest rate
- K: strike price
- sigma: volatility
- T: maturity time
- M: the number of simulations

- N: the number of subintervals

Output

- res: time-0 option price
- stderr: standard error of res

```
function [res,stderr] = LSMC(s0, r, K, sigma, T, M, N)
    % get the length of each subinterval
    delta_t = T/N;

    % initialize the MC stock price matrix: a M * N matrix where
    %  each row represents a stock price path of length N
    %  the stock price is assumed to follow a geometric brownian motion

    % ----------- generate stock price ------------------
    % for each path, generate a vector of standard normal distribution
    stnorm = normrnd(0, 1, [M, N]);
    % time value: start from delta_t, increase delta_t each time, and
    % ends at T
    time_lst = delta_t:delta_t:T;
    % generate stock price path
    mc_s_price = s0 .* exp((r- sigma^2/2) * time_lst + ...
                 sigma * sqrt(time_lst) .* stnorm);

    % ----------- generate stock price ------------------

    % ----------- generate payoff -----------------------
    % return the inner-value of American option
    mc_payoff = max(K - mc_s_price, zeros(M,N));
    % ----------- generate payoff -----------------------

    % ----------- obtain value --------------------------
    % initialize the value of option matrix:
    %  a M * N matrix where each row represents
    value_mat = zeros(M, N);

    % at time N, the option value of option is
    %  K - S_T if K > S_T
    %  0 otherwise
    value_mat(:, N) = mc_payoff(:, N);

    % at time < N, we need to compare between
    %  X: the stock price at time j   and
    %  Y: the discounted cash flow received at time j+1
    for j = N-1:-1:1
        % get the polynomial: polyfit(X = price at column j, Y = value at j
        % + 1 discounted, degree = 4)
        po = polyfit(mc_s_price(:,j), value_mat(:,j + 1) * exp(-r), 4);
```

```
        % step 2: find 'continuation value',
        % obtained by evaluating the polynomial at stock
        % price at t
        cont = polyval(po, mc_s_price(:, j));

        % step 3: compare 'exercise' and 'continuation' columns
        % compare result. If 'exercise value' = payoff at column t >
        % continuation value, then we use payoff. Otherwise, we discount
        % the value
        condition = mc_payoff(:,j) > cont;
        value_mat(condition,j) = mc_payoff(condition,j);
        value_mat(~condition, j) = value_mat(~condition, j+1) * exp(-r);
    end

    % get the option value at time 0
    op_val = value_mat(:, 1) * exp(-r);

    % result is the mean of op_val
    res = mean(op_val);
    stderr = std(op_val)/sqrt(M);


end
```