QF4102 Assignment 2
Choi Yat Long, Low Jia Jun

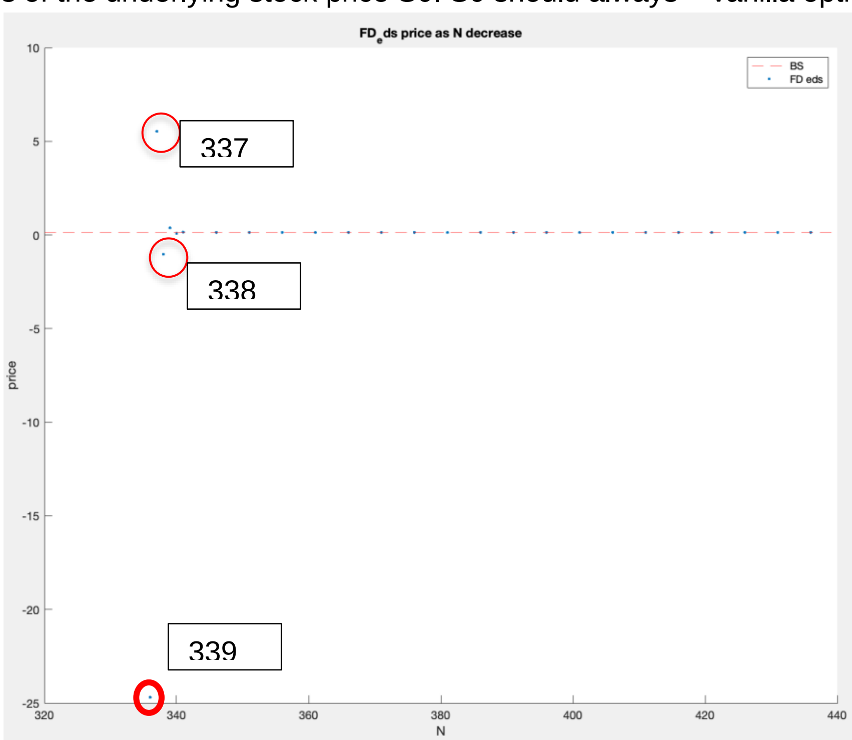
Question 1

(ii) With the input parameters $S_0 = X = 1, T = 0.5, r = 0.02, \sigma = 0.5, q = 0.03$
For $\triangle t = 0.01$ and $h = 0.05$, the Matlab function FD_eds_call gave a price of -5.61011e20 and the exact BS price is 0.1361.

(iii) The lower bound of $N = \frac{T}{\triangle t}$ such that all coefficients are nonnegative is determined to be 436.

(iv) Using the lower bound of 436 and Matlab function FD_eds_call gives a v_fd_eds of 0.1358 which is slightly less than exact BS price of 0.1361. Such a difference can be explained by truncation error of $S_{max} = 3 \times X$ in FD_eds_call. As N increase further, the v_fd_eds is still 0.1358.

(v) For N = 338, option estimates become meaningless as option price estimates become negative. When N = 337, although the value > 0, it is still meaningless as the price is around $5 which is 5 times of the underlying stock price S0. S0 should always > vanilla option price.



Question 2

(i)

Algorithm 1 Two-state-variable forward shooting grid method for a fixed-strike arithmetic Asian call option

Inputs: $S_0, X, r, T, \sigma, q, N, L$

Precompute Constants:

$\Delta t = \frac{T}{N}$ ▷ Time step size

$\Delta x = \sigma\sqrt{\Delta t}$ ▷ Log step size

$u = e^{\sigma\sqrt{\Delta t}}$ ▷ Up-move factor

$p = \frac{e^{(r-q)\Delta t} - u^{-1}}{u - u^{-1}}$ ▷ Risk-neutral probability

$\rho = \frac{1}{L}$

Terminal Condition:

for $k = -LN$ to LN **do** ▷ Loop over average grid

  $A_N^k = S_0 \exp(\rho k \Delta x)$ ▷ Arithmetic average at maturity

end for

for $i = 0$ to N **do** ▷ Loop over stock price grid

  for $k = -LN$ to LN **do**

    $V_N^{k,i} = \max(A_N^k - X, 0)$ ▷ Option payoff at maturity

  end for

end for

Backward Iterations:

for $n = N - 1$ to 0 **do** ▷ Backward time iteration

  $V_n = V_{n+1}$ ▷ Initialize value at current time step

  for $i = n$ to 0 **do** ▷ Loop over stock price grid at time n

    $S_n = S_0 u^{2i-n}$ ▷ Stock price at time step n

    for $k = -Ln$ to Ln **do** ▷ Loop over average grid at time n

      $A_n = S_0 \exp(\rho k \Delta x)$ ▷ Arithmetic average at time step n

      Compute arithmetic average updates for up and down moves:

$$A_{n+1}^u = \frac{S_n u + (n+1)A_n}{n+2}$$

$$A_{n+1}^d = \frac{S_n u^{-1} + (n+1)A_n}{n+2}$$

      Compute grid indices for up and down movements:

$$k_u^{n+1} = \frac{\ln(A_{n+1}^u / S_0) \cdot L}{\Delta x}, \quad k_d^{n+1} = \frac{\ln(A_{n+1}^d / S_0) \cdot L}{\Delta x}$$

      Obtain the corresponding values from the next time step:

$$V_{n+1}^{k_u^{n+1}, i+1}, \quad V_{n+1}^{k_d^{n+1}, i}$$

      Interpolate to get the value at the current time step:

$$V_n^{k,i} = e^{-r\Delta t}\left(p \cdot V_{n+1}^{k_u^{n+1}, i+1} + (1-p) \cdot V_{n+1}^{k_d^{n+1}, i}\right)$$

    end for

  end for

2

end for

Output:

The option value at time $t = 0$ is $V_0^{0,0}$

(iii) With the input parameters $S_0 = X = 100, r = 0.03, T = 1, N = 4, L = 2, \sigma = 0.22, q = 0$

The Matlab function fsg_fixArithAsianCallNew gave a price of 5.34.

(v)

Tabulate option value for $N = 60, 120, 180, 240$ for $\rho = 1, 0.5, 0.25$:

Option Value Estimates:

	Rho_1	Rho_0_5	Rho_0_25
N_60	11.431	10.683	8.4904
N_120	11.59	11.291	9.7889
N_180	11.615	11.502	10.573
N_240	11.558	11.409	10.852

Comment: Historical average of underlier already makes the Asian option in-the-money (historical average $>$ strike). In general as $\rho = 1 \rightarrow \rho = 0.25$, the option value estimates ↓, one possible reason is because of the reduced sensitivity of the averaging process to the underlier price movements as $\rho = \frac{1}{L}$, larger L means we consider more arithmetic average spacing step which leads to higher precision when we are averaging. In particular, more grid points will be below X at maturity.

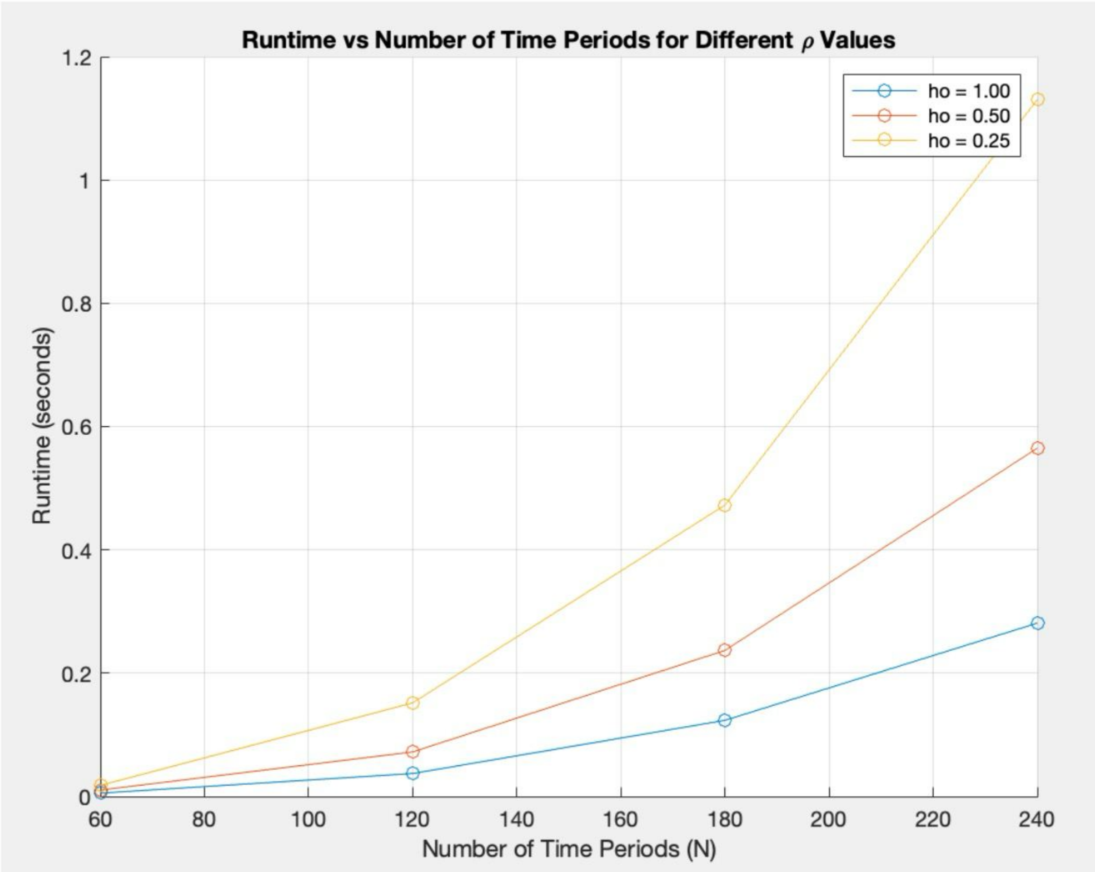
Tabulate runtime for each value of N and ρ:

Runtimes (seconds):

	Rho_1	Rho_0_5	Rho_0_25
N_60	0.051167	0.027799	0.021859
N_120	0.038972	0.07065	0.14218
N_180	0.12317	0.23942	0.46894
N_240	0.29329	0.56526	1.1387

Comment: As N gets larger, for each $\rho_1, \rho_{0.5}, \rho_{0.25}$, the runtime generally increases.

(vi) Plot $runtime$ versus N:



Comment: Compared to assignment 1 where the BTM algorithm grows exponentially $O(2^N)$, which is often very bad and inefficient, the computational efficiency of FSG method grows polynomially which is a huge improvement* in efficiency compared to the BTM algorithm in assignment 1 (i.e. $O(Ln^2)$ for FSG method vs $O(2^N)$ for BTM method) *for large values of N.

The choice of L: As $L\ \uparrow$, though we are sacrificing more compute time, a higher L will give more accurate estimates in option value because there are more log time steps for discrete arithmetic averages samples.