QF4102 Assignment 2 Report

Tan Ming Hui (A0239924U), Jin Leyi (A0294450A)

October 23, 2024

1 Question 1

1.1 (i)

FD_eds_call.m submitted.

1.2 (ii)

At $S_0 = 1$, the Black-Scholes price obtained is 0.13612 while the FD estimate gives $-5.601119565408952 \times 10^{20}$. This estimation not only loses all the significant figures, but is also highly inaccurate and unacceptable. The estimate possesses a significant negative value whilst Black-Scholes gives a positive value. Checking for violations of the monotonicity condition, we get the following:

- Coeff a, Of 59 elements, 0 violated the positivity condition.
- Coeff b, Of 59 elements, 39 violated the positivity condition.
- Coeff c, Of 59 elements, 0 violated the positivity condition.

Clearly, the coefficient b has positivity condition violated thus making the scheme non-convergent. As we can see, the estimated value we got is too far away from the true value. This is because the N we use is too small. This this case, the explicit scheme becomes non-convergent.

1.3 (iii)

From the formulation of EDS(II), coefficients a and b might not, in all cases, satisfy monotonicity conditions. That is, a and b might not always be positive. From question, we get:

$$r = 0.02, \quad q = 0.05, \quad \sigma = 0.5, \quad T = 0.5, \quad h = 0.05, \quad S_0 = X = 1$$

$$i_{max} = \frac{3X}{h} = \frac{3}{0.05} = 60, \quad \Delta t = \frac{T}{N} = \frac{0.5}{N}$$

$$a = \frac{1}{1 + r\Delta t}[\frac{1}{2}(\sigma^2 i^2 - (r - q)i)\Delta t]$$

$$b = \frac{1}{1 + r\Delta t}(1 - \sigma^2 i^2 \Delta t)$$

Since Δt is always > 0, given values in the question, we can see that coefficient a will always be positive. Now, we only need to check for positivity of b. By substituting whatever is known, we get:

$$\frac{1}{1 + r\Delta t}[1 - \sigma^2 i^2 \Delta t] > 0$$

$$[1 - (0.5)^2(60)^2\frac{0.5}{N}] > 0$$

$$N > 450$$

We get $N_{min} = 451$.

1.4 (iv)

Changing N to 451, keeping all others the same, the estimated value improves to 0.13583. Comparing with the Black-Scholes price which is 0.13612, we can see accuracy have significantly improved. The first two significant figures are the same now.

1.5 (v)

By gradually decreasing the N value, we obtain the following estimates:

N	Exact Value	FD value
345	0.13612	0.13586
344	0.13612	0.13574
343	0.13612	0.13631
342	0.13612	0.13356
341	0.13612	0.14687

Table 1: Finding cutoff value

As we can see, the cutoff value is $N = 341$, where the estimated value is 0.14687, where all significant figures do not tally with the exact value. 341 is smaller than the previous lower bound obtained, which is 451. This demonstrates the calculated lower bound is sufficient but not necessary for getting at least one significant figure in the estimate to tally with the exact Black-Scholes price.

2 Question 2

2.1 (i)

Fixed Arithmetic Asian Call Option Pricing Algorithm (NEW)

Initialize Average vector

$$Average_k = S_0 e^{k \cdot dy}, \quad k = -NL, -NL+1, \ldots, NL$$

where k represents the index for the average.

for $j = 0, \ldots, N$ **do**
  for $k = -NL, \ldots, NL$ **do**
    $V[j+1, k+kshift] = \max(Average[k+kshift] - X, 0)$
where $kshift$ = NL+1.

for $n = N-1, N-2, \ldots, 0$ **do**
  Initialize $Vtemp$ array of size $(n+1) \times (2nL+1)$
  for $j = n, n-1, \ldots, 0$ **do**
    $S_n^j = S_0 e^{(2j-n)dx}$
    for $k = -nL$ **to** nL **do**
      $A^k = Average[k+kshift]$

      Get k index for up branch
      $A^u = \frac{S_n^j u + (n+1)A^k}{n+2}$
      $k^u = \frac{\log(A^u/S_0)}{dy}$
      Translate to index for avg vector
      $ku_{fl}^a = floor(k^u) + kshift$
      $ku_{cl}^a = ku_{fl}^a + 1$
      Translate to index for V matrix
      $ku_{fl}^v = floor(k^u) + (n+1) * L + 1$
      $ku_{cl}^v = ku_{fl}^v + 1$
      Interpolate $V(j+2, ku_{fl}^v)$, $V(j+2, ku_{cl}^v)$ for V^u

      Get k index for down branch
      $A^d = \frac{S_n^j d + (n+1)A^k}{n+2}$
      $k^d = \frac{\log(A^d/S_0)}{dy}$
      Translate to index for avg vector
      $kd_{fl}^a = floor(k^d) + kshift$
      $kd_{cl}^a = kd_{fl}^a + 1$
      Translate to index for V matrix
      $kd_{fl}^v = floor(k^d) + (n+1) * L + 1$
      $kd_{cl}^v = kd_{fl}^v + 1$
      Interpolate $V(j+1, kd_{fl}^v)$,$V(j+1, kd_{cl}^v)$ for V^d

Update Vtemp matrix
$Vtemp(j+1, k+(n*L)+1) = e^{-rdt}(pV^u + (1-p)V^d)$

Update: $V = Vtemp$

Output: $V(1,1)$

2.2 (ii)

fsg_fixArithAsianCallNew.m submitted.

2.3 (iii)

$r = 0.03, \quad q = 0, \quad \sigma = 0.22, \quad T = 1, \quad N = 4, \quad L = 2, \quad S_0 = X = 100$

Output: 5.7397

2.4 (iv)

fsg_fixArithAsianCall.m submitted.

Key modifications:

1. Change in Average vector initialization
   (a) For newly issued options, initial runavg is S0. Now, we have to update runavg using the arithmetic average formula to incorporate historical averages: $(runavg * Nhist + S_0)/(Nhist + 1)$
   (b) Range of final average also changes according to current runavg: $S_0 e^{kdy}$ becomes $runavg * e^{kdy}$ for $k = -NL, -NL+1, \ldots, NL$
2. Change in average calculation
   (a) We need to incorporate the previously elapsed periods when calculating averages, specifically
   (b) $A^u = (S_n^j u + (n + Nhist + 1)A)/(n + Nhist + 2)$
   (c) $A^d = (S_n^j d + (n + Nhist + 1)A)/(n + Nhist + 2)$
3. Position of the averages
   (a) We need to incorporate runavg into calculation of k^u and k^d
   (b) $k^u = log(A^u/runavg)/dy$
   (c) $k^d = log(A^d/runavg)/dy$

2.5 (v)

N	$\rho = 1$	$\rho = \frac{1}{2}$	$\rho = \frac{1}{4}$
60	6.6752	6.5703	6.5149
120	6.6142	6.5396	6.5020
180	6.5819	6.5254	6.4964
240	6.5717	6.5176	6.4926

Table 2: Estimated Option Values

Comment on option values:

- Keeping ρ constant, as N increases, the estimated option value decreases at a decreasing rate.
- Keeping N constant, as ρ decreases, the estimated option value decreases at a decreasing rate.
- It seems like there is a convergence as N increase and ρ decrease due to the option values decreasing at a decreasing rate. This indicates some form of convergence.

N	$\rho = 1$	$\rho = \frac{1}{2}$	$\rho = \frac{1}{4}$
60	0.0156	0.0312	0.1406
120	0.2344	0.4375	0.8281
180	0.6875	1.2969	2.5781
240	1.0938	2.3906	4.3125

Table 3: Function runtime

Comment on runtime:

- As ρ is divided by k, meaning the number of grids is multiplied by k, holding N constant, the runtime is multiplied by k.
- As N increases, keeping ρ constant, the runtime increases more than n does. In particular, time complexity seems to range from $O(n^2)$ to $O(n^3)$.

2.6 (vi)

Comment:

- The time complexity for FSG method is theoretically $O((n+1)(\frac{2n}{\rho}+1))$. This means that the runtime will increase slightly more than n^2 as the number of time periods increase.
- The time complexity for BTM method is $O(2^n)$. This shows an astonishing exponential runtime, which is very inefficient.
- Specifically, as can been seen from the following figures, the BTM method needs approximately 0.425 runtime for just N = 20, while FSG method only takes 0.1406 for N = 60, $\rho = \frac{1}{4}$.
- This shows that the FSG method is much more efficient than BTM, and is especially so as N increases.

Figures on next page: Figure 1 shows FSG runtime for **Fixed Arithmetic Asian Call**. Figure 2 shows BTM runtime for **Fixed Geometric Asian Call** from last part of assignment 1, both European.

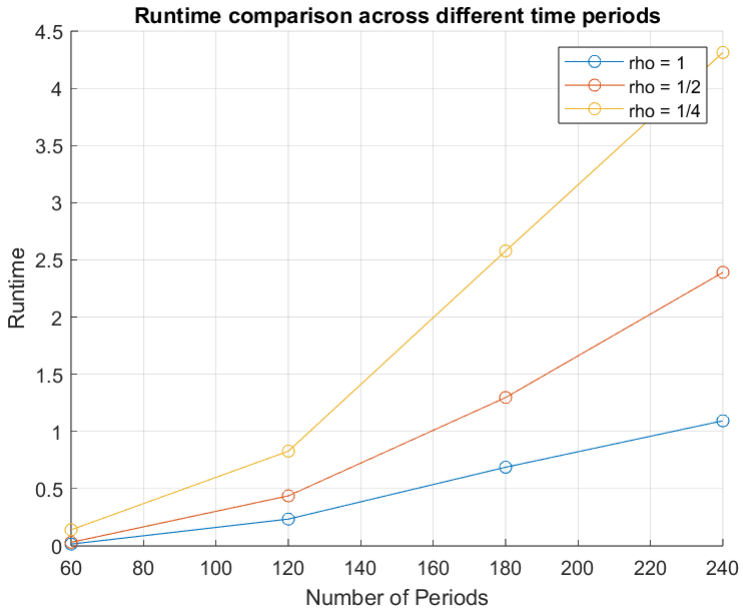

Figure 1: FSG runtime

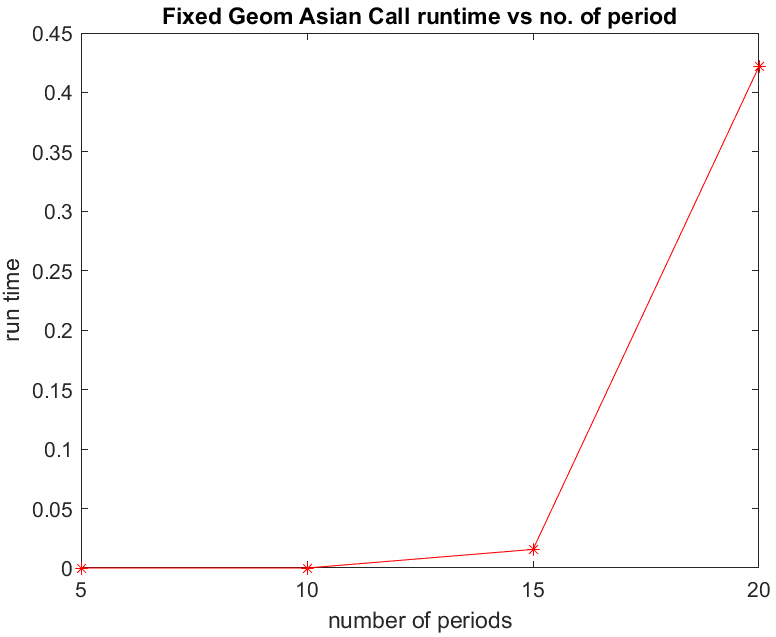

Figure 2: BTM runtime