Introductory Computational Finance

Kaiki Ikeda

QUANTNET Baruch PRE-MFE C++ for Financial Engineering

May 2025

Group F: Finite Difference Methods (Introduction)

a) The code was arranged so it can run the FDM test. Paths were all adjusted accordingly and all excel files were outputted.

b) After I got the code to run, I tested batches 1-4 and analyzed each output. I noticed how these values changed with respect to the relationship of N and J values. For batch 1 testing, I started off with N = 499 and J = 5 * BS::K and the accuracy and speed was good. It yielded a numerical FDM price of 5.84254, which has two places after the decimal place being identical to the exact solution. I then wanted to see if I can improve the accuracy of the FDM output so I gradually increased N to 9999 and kept J the same, this resulted in an output of 5.82207 which didn't make much of a difference. Lastly, I decided to set N to 49999 and J = 10 * BS::K to accommodate for the size proportionality; this yielded a result much closer to the exact solution.

For batch 2 testing, I started off with N = 9999 and J = 5 * BS::K which gave me an output of 7.96321. I wanted to get closer to the exact solution 7.96557 so I increased N to 99999 but it yielded an output farther off the exact price. I found that increasing N too much without increasing J can make results worse due to the grid mismatch or round off accumulation. FDM code uses an explicit method where stability depends on:

change in t < C * (change in x) ^ 2

It seems that a disproportionately large N results in too many tiny steps. Furthermore, if J is not fine enough, the spatial resolution becomes stuck.

I ended batch 2 testing with N = 99999 and J = 10 * BS::K and the output was 7.96496. This proved my speculations and allowed me to compute better values for higher precision output.

Batch 3 testing was performed starting with N = 9999 and J = 5 * BS::K, which gave me an output with a precision of one place after the decimal. I decided to increase J to 10 * BS::K and it resulted in better precision when comparing it to the exact solution.

Lastly, batch 4 testing yielded an error value for me. I was very confused to see -nan(ind) at first but after further observations, I was able to notice some things. This scenario was an edge case due to the fact that this batch has a very large time to expiry of 30 years. I assume when we measure change in T, we are doing 30/N, which is a very small number. We also have to be careful of the stability condition. In this case, our change in T is very likely to be greater than C * (change in x) ^ 2. I figured we could fix it by increasing N much more and it outputted a valid value (1.19586).