**FDM option APPLICATION OVERVIEW**

**Name: William Zhang**

**1.OVERVIEW**

The FDM option pricing application is for pricing various options such Asian, European and Barrier Option using various Finite Difference method. The application allows the user to input various Option pricing parameter such as stock price, strike price, risk-free interest rate and so forth to price options using the various Finite Difference Method. The application even allows the user to use various Random number generators such as Mersenne Twister, Polar Masglia and Bob Mueller. Finally, the user can input the data by console or by putting the data inside the input.csv. If the user wishes to put more than one option dataset parameter, then the user should put the data into the input

**2.CLASS OVERVIEW**

The user can see the class diagram hosted inside in the Solution Explorer, by opening the project solution folder and view ClassDiagram.cd. The following is a description of the various classes.

**1.RNG**

This class essentially generates a random number that follows a normal distribution with mean zero and variance one. The user can select between various generators such as Mersenne Twister, Polar Masglia and Box Mueller. The latter two transforming uniform random numbers from zero to one to a normal distribution with polar form

**2.SDE**

This class specifies which types of stochastic differential equation. The user is able to choose between the contrast elasticity variance model (CEV) and the Geometric Brown Motion Differential equation (GBM). The former being like GBM but there is a specified Beta that model the elasticity of volatility in relation to stock price.

**3.FDM**

This class solves the Stochastic differential equation provided by allowing the user to choose the Finite difference method to solve the differential equation. The user can use choose explicit Euler, Balanced Midpoint, and Milstein Method to solve the differential equation

**4. PRICER**

This class models the various types of options that the user can choose. Different options have different payout schemes. The user can choose between the various option such as the classical European, The Asian, or Barrier Options.

**5. BUILDER**

This class essentially coalesces all the various the parts the user want and builds the appropriate RNG,SDE,PRICER,and FDM objects to start the Finite Difference method application. The parts are created and stored into a tuple

**6.MEDIATOR**

The parts made from the Builder class must be mediated because the parts made from Builder have complex interactions. Therefore, we have a mediator to resolve these interactions inside a STRUCT to generate an output. The LOOP in the start function is parallelized with Pragma OMP to improve speed.

**7.FILEIMPORTER**

This header file is the application input point and is by far the largest file. This file allows the user the start the application and allows the user to choose input data by console or by File. The user also must specify the #NSIM and #NUMBER of trials as well as which parts they want in the application. This is where the mediator does most of its work.

**3.COMPLEXITY/PERFORMANCE/OUTPUT**

Most of the work performed is the double for loop inside the Mediator start function. So in general the time complexity of this program is **O**(NSIM\* SUBDVISIONS). Which is quadratic and in general has a convergence that is linear. For inputs NSIM\*subdivisions >= 5,000,000 performance starts slowing down. Therefore we incorporated multi-threading that is provided by the pragma OMP

//Across all applications and option parameter inputs. For NSIM = 1,000,00 and subdivisions = 500

**//As evidence for the time, check the output and check the option data input.**

**WITHOUT MULTITHREADING:**

**Time to completion : 90.6-98 seconds**

**NSIM: 1000000 NT:500**

**Total time: 87.0217**

**Output # 1 Price: 5.71458**

**Total time: 92.582**

**Output # 2 Price: 2.2088**

**Total time: 98.522**

**Output # 3 Price: 7.7895**

**Total time: 90.39**

**Output # 4 Price: 94.1845**

**Total time: 89.9149**

**Output # 5 Price: 1.22124**

**Total time: 89.6994**

**Output # 6 Price: 4.04489**

**Total time: 98.1227**

**Output # 7 Price: 0.209067**

**With Multithreading (OMP)**

**Time to completion: 32-42 seconds**

**NSIM: 1000000 NT:500**

**Total time: 29.0217**

**Output # 1 Price: 5.71458**

**Total time: 31.5582**

**Output # 2 Price: 2.2088**

**Total time: 32.5722**

**Output # 3 Price: 7.7895**

**Total time: 34.639**

**Output # 4 Price: 94.1845**

**Total time: 34.9149**

**Output # 5 Price: 1.22124**

**Total time: 36.6994**

**Output # 6 Price: 4.04489**

**Total time: 37.4527**

**Output # 7 Price: 0.209067**

As you can see here the performance was increased by 50 percent with OMP parallel. The user can choose his own and parameter schemes to check the accuracy of the application. I also tested the version with Asian options and maintained 2-3 cent accuracy. And here is CD provided by the MSVC

Diagram, schematic

Description automatically generated