Queens College, CUNY, Department of Computer Science
Computational Finance
CSCI 365 / 765
Fall 2017
Instructor: Dr. Sateesh Mane

Equity Derivatives Library Project "Discussion Board"

October 16, 2017

1 Introduction

- This document is intended as a "forum" or "discussion board" to exchange ideas.
- Hence this document will grow with time, and you should check it for news and updates.
- Send an email to me with your ideas or suggestions.
- For example one suggestion was to reclassify this as a "project" to separate it from the homework assignments. It was correctly pointed out that this is a multi-week project and is not connected to a single topic or lecture.
- Participation in the project is voluntary.
    1. The project was not mentioned in the syllabus, so there is no official grade for it.
    2. It is an opportunity for you to ***design a program***.
    3. Not just to write a program for something that a lecturer/textbook invented.
- You can also send an email with comments on material which has been posted.
- I will examine each submission and possibly edit it for content.
- Contributions will be posted by date.
- The ideas/comments can be about many things:
    1. New things to do.
    2. Solution of a currently open problem.
    3. Improvement on an idea which has been posted.
    4. Loophole/bug in an idea which has been posted.
    5. Etc.

2 (10/16/2017) Some thoughts

- This project is an opportunity for you to understand what it means to create a "user friendly" interface.
- To do that, we need to think about what a user (client) would want to do.
- One obvious task which all clients will want is to calculate the fair value of a financial security.
  1. Hence let us use that as our starting example.
  2. We have one "high level" function at present, and that is `FairValue`.
  3. If we find a "good" way to implement `FairValue`, that will probably serve as a "template" or "paradigm" for other high level functions.
- What issues do we face so far to implement `FairValue`?
  1. A client could forget to populate the indicatives data and/or market data before calling `FairValue`.
  2. We could solve the above problem by expanding the function signature of `FairValue` so that the calling application must input an `IndicativeData` and a `MarketData` object when calling `FairValue`. This guarantees that the input data will be available (and can be validated) before doing calculations.
  3. However, the above idea raises the issue: how will the calling application input client overwrites?
- ***Why are client overwrites so important?***
  - As I explained, a client might wish to calculate the fair value of an option in a loop, for $S = 1$ to $S = 100$. Here S is the stock price, and obviously it has no connection to the actual market price of the stock.
- One possible solution is to ask the calling application to overwrite the indicative data and market data objects, so that all the overwrites are already included in the input data before `FairValue` is called.
- This would work, but is it the only possible solution? It is a "user friendly" solution? Clients might find it tiresome to overwrite the data objects. Essentially they would be destroying the original data. This might be unpopular with clients.
- Another possible solution, ***and it is a good idea***, is to create a new `ClientData` class, for the calling application to input overwrites.
- Note that the two ideas are not mutually exclusive. A calling application can always overwrite the input data objects. The `ClientData` class is a bonus (a "user friendly" bonus) to make life easier for the client.
- *But what would the* `ClientData` *class look like? How would you design it?*
- ***Ideas?***