**Python Derivative Analytics Library Project**

(In progress)

*code:*[*https://github.com/RuitaoWang/Derivative\_Analytics/blob/master/Deriv\_Analytic\_Library.ipynb*](https://github.com/RuitaoWang/Derivative_Analytics/blob/master/Deriv_Analytic_Library.ipynb)

*-Ruitao Wang (recent grad in MSAF at Lehigh University)*

*email:* [*ruitao.wang@outlook.com*](mailto:ruitao.wang@outlook.com) *| Cell: (484) 447-9238*

*10th updated edition*

**Summary**: Upon completion, this Derivative Analytics Library should function in all aspect of derivative analytics such as market environment information arrangement, simulation generator, model calibration, and derivative valuation. The development of this library aims to familiarize myself with both mathematic behind derivative valuation models and Objective-Oriented Programming concept. So far I have already learned a lot technical details for this project through self-study from sources like Python package documentation, [www.pythonprogramming.net](http://www.pythonprogramming.net/), and ‘Python for Finance’ by Yves Hilpisch

**Content**

Library Map: (Yellow to be updated) 2

Functionality: 3

Next Step: 4

## Library Map: (Yellow to be updated)

I intend to divide this library into 4 major parts that each was based on a base class with subclass for different categories of demand:

1. ***Market Environment***
2. ***Simulation***
3. ***Single Derivative Valuation***
4. ***Portfolio Analytics***

Market Environment

Date Manipulation

Discount Curve

Visualization

Distribution

VaR

Valuation

Portfolio

Combined with position class

function

Visualization

Metrics

Chose one simulation method

sub

sub

Mean-reverting (CIR)

Jump Diffusion

Geometric Brownian Motion

Take as input

Constant Parameter

Simulation (generic class)

European Option

American Option (Longstaff & Schwartz)

Valuation (generic class)

Other (to be decided)

## Functionality:

1. **Market Environment**

* **get\_YearDelta** (global function): take an array of datetime formatted date and transformed it into an array of day delta in years.
* **const\_short\_rate (class)**:

1. generate 2 dimensional discount ratio matrix for constant spot rate.

* **short\_rate (class)**:

1. generate 2 dimensional discount ratio matrix for non-constant spot rate.
2. get the spot curve with cubic spline interpolation and plot.

* **mar\_env**(generic class): the base class used to hold all relevant data (constant, curve, list)

1. add information for individual category
2. access information for individual category
3. add whole instance (all categories)
4. print all information in organized format
5. **Simulation**

* **sn\_generatotion** (global function):take desired dimension and a bunch of Boolean variable as input and can return standard normal random numbers with or without variance reduction techniques (moment matching, antithetic).
* **simulation\_class** (generic object): abstract class used to format all different simulation. Output was simulated paths as underlying.

1. generate the time grid used for path simulation
2. call path generation method from sub class (GBM, Jump Diffusion, etc)
3. plot the histogram for the final values of paths generated

* **geometric\_brownian\_motion (object):** sub class for GBM simulation

1. update information and clean path for instance
2. generate GBM paths with Euler scheme (Cholesky method for correlated simulations)

* **jump\_difussion (object):**

1. update information and clean path for instance
2. generate jump deffusion paths with Poisson Process enhanced Euler scheme (Cholesky method for correlated simulations)

* **mean\_reverting (CIR)(object):**

1. update information and clean path for instance
2. generate mean reverting paths with positive enhanced Euler scheme (Cholesky method for correlated simulations)

* **other to be brainstormed**

1. **Valuation**

* **valuation\_class (generic class):** generic valuation class for all classes

1. update information
2. delta calculation (effective delta)
3. vega calculation (effective vega)

* **valuation\_mcs\_euro** (Object): sub class for European Option valuation with Monte Carlo.

1. generate payoff along the simulated paths
2. calculate option value

* **longstaff\_schwartz** (object): sub class for American Option valuation with with Longstaff\_schwartz model.

1. Option valuation
2. **Portfolio**

* **derivative\_position** (generic class): **t**he base object holding market environment and position data.

1. print portfolio information
2. calculate portfolio statistics

* **option derivative position (to be updated)**

## Next Step:

For next step, I’d like to firstly add more sub class methods and functionalities for different derivatives (such as bond, interest rate model, etc). After abovementioned is finished, I’d like to add the calibration module, which I believe should be the toughest part of this library since it involves not only analytics, but also big dataset manipulation.

**For more details please refer to:**

[*https://github.com/RuitaoWang/Derivative\_Analytics/blob/master/Deriv\_Analytic\_Library.ipynb*](https://github.com/RuitaoWang/Derivative_Analytics/blob/master/Deriv_Analytic_Library.ipynb)