**Assignment 2 Yield Curve Calibration**

# **Introduction**

This assignment follows market standard practice of computing XVA by Monte Carlo simulation. Students are expected to achieve the following:

1. Study swap XVA example, and understand the valuation framework: model initialization, model evolution, exposure matrix computation, and aggregation with credit/funding component.
2. Build FX Geometric Brownian Motion (GBM) model with stochastic interest rates, evolve the model and simulate FX spot rate
3. Understand the drivers behind XVA, model vol, correlation, credit spread etc

**Task 1:**

Download python project CVA from assignment folder. Run *compute\_swap\_xva()* in XVA.py, the sample code computes XVA per a 10Y pay fixed rate interest rate swap. Interest rate is driven by Hull White short rate model, 5000 simulation paths and 40 time steps. XVA valuation follows a few key steps:

1. Gather inputs, curves and simulation related configs
2. Initiate models, HW and GBM model, set simulation slice index to 0, initialize model parameters
3. Compute exposure matrix per time step and per simulation path, this is core component in XVA computation. In method *compute\_exposure\_matrix*, we evolve model from previous time step to current time step, and simulate short rates and fx spots; then compute mtm of the trade as of current time step with simulated curve and fx spot.

Note, our implementation is efficient, we apply vectorized operation in numpy. Thus we never loop path explicitly.

1. Aggregate exposure metric (EDPE, EDNE) with funding/credit curve, and compute final result cva, dva, fca, fba. (You have to complete the aggregation logic in task 2)

No action is required for Task 1, review and understand XVA valuation framework. The trick is in XVA valuation, instead of calling static curve, it calls simulated curve, thus the discount factor is generated from HW model.

**Task 2:**

In XVA.py, complete ***compute\_xva*** function implementation, where function prototype is defined in *compute\_swap\_xva,* commented out by default. After implementation, uncomment the last few lines from *compute\_swap\_xva()* method, and print out cva, dva, fca, fba. Your result should be close to cell K12:K13 in sheet “17.2 and 17.3” of 4E-Chapter-17.xlsm, K9 set to false.

**Task 3:**

FX spot rate under GBM model is formulated in this way:

Where is a standardized normal random variable, and is short rate from domestic currency HW model, and foreign currency short rate model respectively.

In a discrete simulation set up, from previous simulation time s to current time t, we can rewrite the formulae:

In XVA.py main method, comment out *compute\_swap\_xva*, uncomment *compute\_fxfwd\_xva*, implement ***simufxspot***class in *SimuFXSpot.py*, 4 functions (roughly 30 lines of code). The function prototypes can be found in *SimuFXSpot.py*

Compute cva and fca, and compute distribution of fx spot rate at final time step (5Y): 2.5%, mean, 97.5%.

**Task 4:**

Understand the driver of cva and fca:

1. Change fx vol from 0.1 to 0.15, compute cva and fca; change dom HW vol and for HW vol from 0.01 to 0.015 respectively, compute cva and fca
2. Original correlation matrix is an identity matrix, 3 assets are intendent. Apply 2 different correlation matrices respectively, and compute cva and fca

np.array([[1, 0.5, -0.5], [0.5, 1, 0.5], [-0.5, 0.5, 1]])

np.array([[1, -0.5, 0.5], [-0.5, 1, -0.5], [0.5, -0.5, 1]])

Hint: take variance of

**What o submit:**

* 2 Python scripts, XVA.py and SimuFXSpot.py, I replace your files in my folder, I could print results in console for FXForward, e.g., print(‘cva’, results\_dict[‘cva’]) will work
* In a one-page simple report, print a table: the base cva and fca result is from task 2, then 5 sets cva and fca results from task 4
* Compare and discuss the impact of FX vol vs IR vol (task 4) to CVA.
* Discuss the impact of risk factor correlation to CVA.