Vanilla Fixed Income Library – KIRA

Author: Lun Li

FE-522 Final Project

Dec 6th, 2019

Comments

Warning: Author is not Financial Engineering Major. Please bear with me and forgive my naivety...

Acknowledgement: This project cannot be accomplished without discussions with my friend who is a desk quant in major investment bank. He gives me a lot valuable suggestions in terms of library design as well as fixed income derivatives knowledge.

Outline of the presentation:

- Objective and Products and Models Coverage
- Design Philosophy and Overview
- A closer look: component by component
- Demo A Vanilla Desk with Excel Interface

Objective/Product and Model Scope

Target

Design a mini-version of library that can support a fixed income vanilla desk, i.e., pricing linear products and volatility products.

Product Coverage

Products:

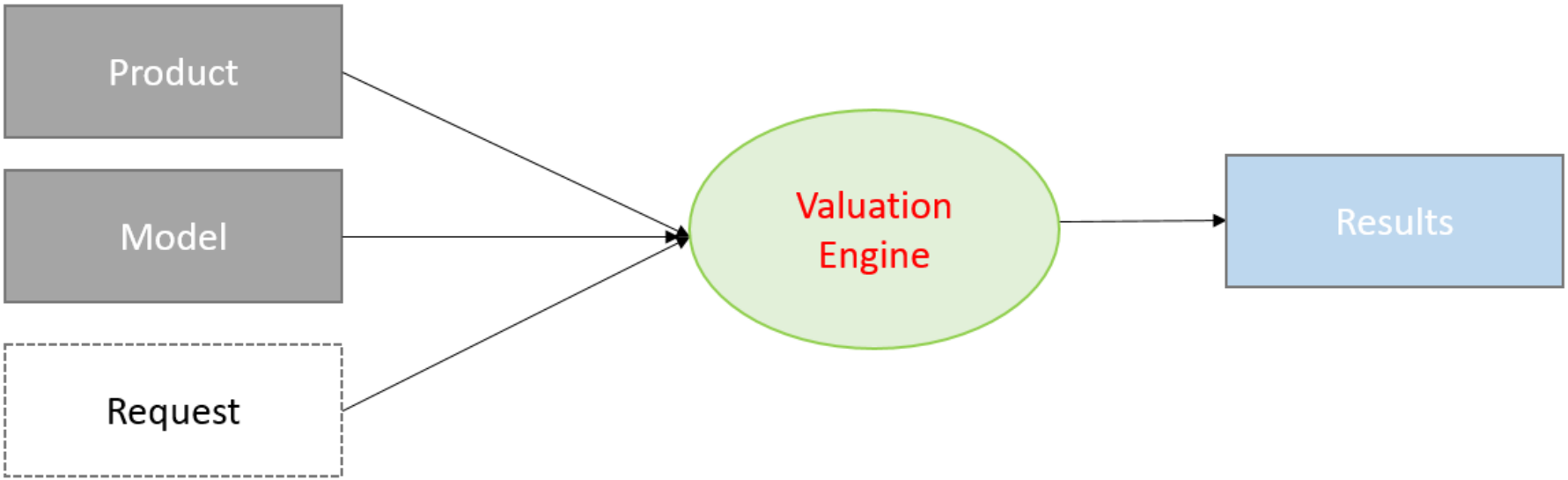
- Cash Deposit (linear)
- Forward Rate Agreement (liner)
- Interest Rate Swap (linear)
- Cap/Floorlet (non-linear)
- Swaption (non-linear)

Model Coverage

- Bachelier Model – Normal Volatility Model;
- Stochastic Alpha Beta Rho (SABR) Model (Industrial Standard).

Design Overview

Main components of Library is model, product, valuation engine.



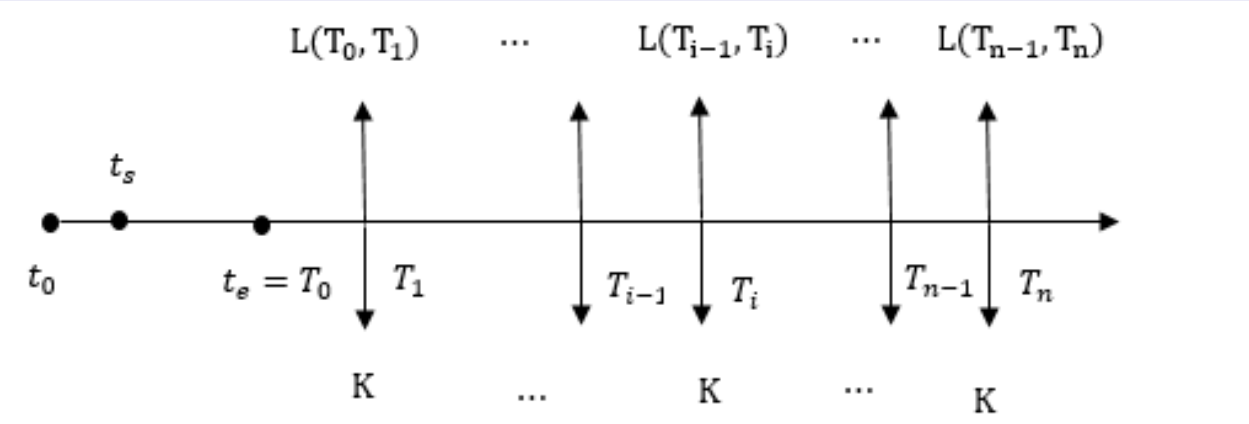
Remark

The interaction is all happening inside valuation engine, though they have their own responsibilities, respectively. The request is made by user to trigger corresponding calculation.

Product is a abstract class, from which all specific ones are derived. A product object is a representation of its all attributes.

Example: Swap

Swap is defined by pay or receive, notional, expiry, tenor, fixed rate, swap convention.



The ProductSwap class provides getter function to fish out these information, in its constructor it also generate schedule and save it.

Model

Model consist of unique name, value date, model type, data collection, build method.

- **Unique Name/Value Date** These are trivial;
- **Model Type** indicates if it is yield curve model, or Bachelier model, SABR model;
- **Data Collection** It is a collection of data objects, e.g., market data, stored in a map (each of them of unique identifier), data shape can be 1D vector or 2D matrix (encapsulated in Data Class)
- **Build Method** It is the recipe to build model, essentially a name-value pair objects.

Remark

Model is a abstract class, we make basic functions virtual, e.g., *getModelValueDate()*, e.t.c.. Yield curve and volatility model are both derived class.

Model Construction

It is done by looping through build methods items to identify corresponding data object from data collection and execute calibration.

Example: Yield Curve Model

YC Build Method	
Target	USD-LIBOR-BBA-3M
CASH_DEPOSIT	CASH_USD
FRA	USD-LIBOR-BBA-3M-FRA
SWAP	USD-SWAP-SEMI-BOND
INTERPOLATION	PIECEWISE_CONSTANT_LEFT

Data Collection	
CASH_DEPOSIT-CASH_USD	Data1D
FRA-USD-LIBOR-BBA-3M-FRA	Data1D
SWAP-USD-SWAP-SEMI-BOND	Data1D

Data Objects	
1D	1.53%
1M	1.70%
3M	1.91%
6M	1.90%
9M	2.00%
1Y	1.71%
2Y	1.58%
5Y	1.57%
10Y	2.00%
15Y	2.21%
20Y	2.24%

Valuation Engine

Valuation engine is where everyone meets! What does valuation engine do?

- Ask product to get product information;
- Based on product information, ask model to provide corresponding financial quantities (e.g., discounting factor, forward rate, volatility);
- Assemble all calculations to PV, get break-even rate, PV01, e.t.c..

Therefore, it is abstracted as follows:

```
class valuationEngine
{
public:
  virtual void calculateValue() = 0;
  virtual string getValuationEngineType() const = 0;
  virtual double parRateOrSpread() = 0;
  virtual double pv01() const = 0;
  virtual double value() const = 0;
  virtual double normalVol() const = 0;
};

class valuationEngineCashDeposit { ... };

class valuationEngineFRA { ... };

class valuationEngineSwap { ... };

class valuationEngineCapFloorLet { ... };

class valuationEngineSwaption { ... };
```

Interactive Process

Model calibration solves the problem,

$$F(X^I; X^M) = 0$$

where X^I is internal parameters, X^M is market quotes. The functional F is valuation engine. The equation is solved by a solver, e.g., Newton-Raphson.

Yield Curve Construction

In yield curve construction, X^I is instantaneous forward rate, X^M is cash deposit, FRA, Swap par rate. As we choosing them non-overlapping, the calibration becomes solving

$$Ax = b \tag{1}$$

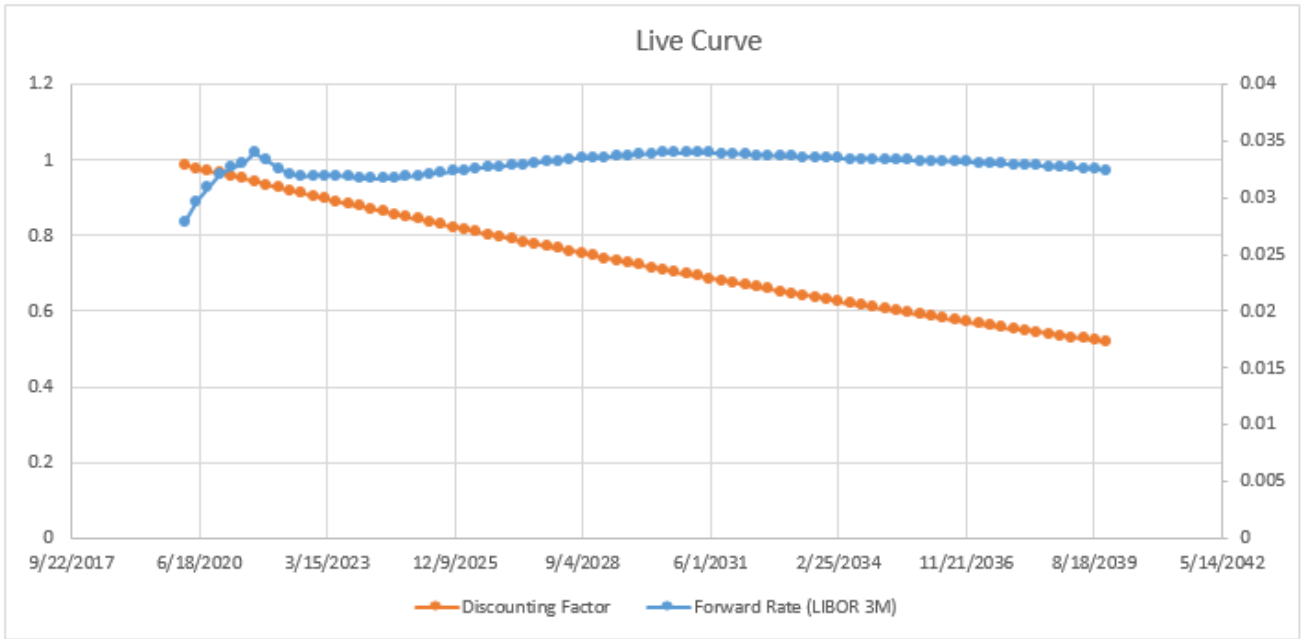
where A is a upper-triangle matrix.

Interface Functions

We have a collection of KIRA API's to be exposed, e.g., kiraCreateModel, kiraCreateProduct, kiraCreateValueReport, e.t.c..

Yield Curve Marking(kiraCreateModelYieldCurve)

	Curve Marking	
Cash	1D	1.91%
FRA	2019-12-18	1.91%
	2020-01-15	1.83%
	2020-02-19	1.76%
	2020-03-18	1.72%
	2020-06-17	1.62%
	2020-09-16	1.54%
	2020-12-16	1.53%
	2021-03-17	1.46%
	2021-06-16	1.45%
	2021-09-15	1.44%
	2021-12-15	1.47%
	2022-03-16	1.47%
	2022-06-15	1.49%
	2022-09-21	1.51%
SWAP	4Y	1.58%
	5Y	1.59%
	6Y	1.61%
	7Y	1.63%
	8Y	1.65%
	9Y	1.68%
	10Y	1.71%
	12Y	1.75%
	15Y	1.80%
	20Y	1.86%



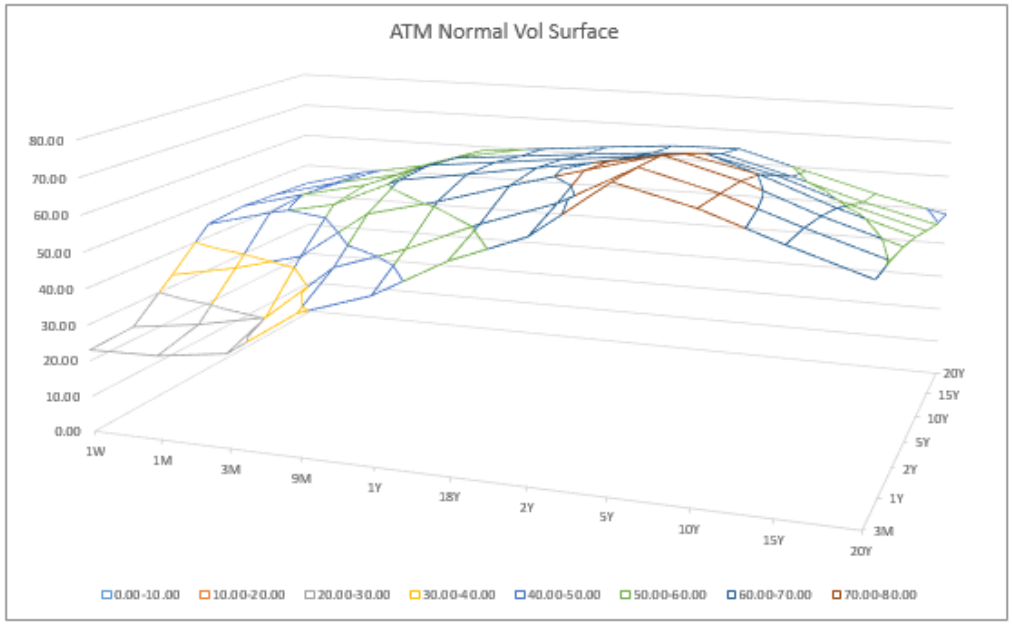
Volatility Marking(kiraCreateVolModel)

Normal Volatility(bps)	3M	1Y	2Y	5Y	10Y	15Y	20Y
1W	23.02	23.59	33.37	43.82	44.94	44.43	43.92
1M	23.47	26.29	37.19	48.83	50.08	49.51	48.94
3M	26.18	30.19	42.30	52.77	54.40	53.45	52.49
9M	39.03	45.94	55.97	61.38	61.77	60.16	58.76
1Y	45.48	52.58	60.89	64.40	63.92	62.00	60.57
18Y	56.14	60.84	66.68	67.61	66.17	63.87	62.39
2Y	63.86	67.72	72.16	70.69	68.24	65.52	64.01
5Y	78.64	78.25	77.55	74.20	70.50	66.58	64.39
10Y	73.83	73.10	72.11	69.82	65.56	61.20	58.88
15Y	66.87	66.20	64.58	62.23	58.71	55.24	53.38
20Y	60.44	59.84	58.38	56.12	53.07	50.24	48.73

Beta	3M	1Y	2Y	5Y	10Y	15Y	20Y
1W	40%	40%	40%	40%	40%	40%	40%
1M	40%	40%	40%	40%	40%	40%	40%
3M	40%	40%	40%	40%	40%	40%	40%
9M	40%	40%	40%	40%	40%	40%	40%
1Y	40%	40%	40%	40%	40%	40%	40%
18Y	40%	40%	40%	40%	40%	40%	40%
2Y	40%	40%	40%	40%	40%	40%	40%
5Y	40%	40%	40%	40%	40%	40%	40%
10Y	40%	40%	40%	40%	40%	40%	40%
15Y	40%	40%	40%	40%	40%	40%	40%
20Y	40%	40%	40%	40%	40%	40%	40%

NU	3M	1Y	2Y	5Y	10Y	15Y	20Y
1W	1.65	1.65	1.60	1.55	1.55	1.56	1.56
1M	1.65	1.65	1.60	1.55	1.55	1.56	1.56
3M	1.25	1.25	1.20	1.20	1.20	1.21	1.21
9M	0.69	0.69	0.71	0.69	0.69	0.69	0.70
1Y	0.59	0.59	0.59	0.59	0.59	0.59	0.59
18Y	0.49	0.49	0.49	0.49	0.49	0.49	0.49
2Y	0.44	0.44	0.44	0.43	0.43	0.43	0.43
5Y	0.34	0.34	0.34	0.32	0.32	0.32	0.32
10Y	0.30	0.30	0.29	0.29	0.29	0.29	0.29
15Y	0.29	0.29	0.29	0.29	0.29	0.29	0.29
20Y	0.29	0.29	0.29	0.29	0.29	0.29	0.29

Beta	3M	1Y	2Y	5Y	10Y	15Y	20Y
1W	-0.55	-0.52	-0.40	-0.20	-0.08	-0.08	-0.08
1M	-0.55	-0.52	-0.40	-0.20	-0.08	-0.08	-0.08
3M	-0.55	-0.52	-0.40	-0.20	-0.08	-0.08	-0.08
9M	-0.55	-0.52	-0.40	-0.21	-0.09	-0.09	-0.09
1Y	-0.55	-0.52	-0.40	-0.23	-0.10	-0.10	-0.10
18Y	-0.53	-0.50	-0.39	-0.24	-0.12	-0.12	-0.13

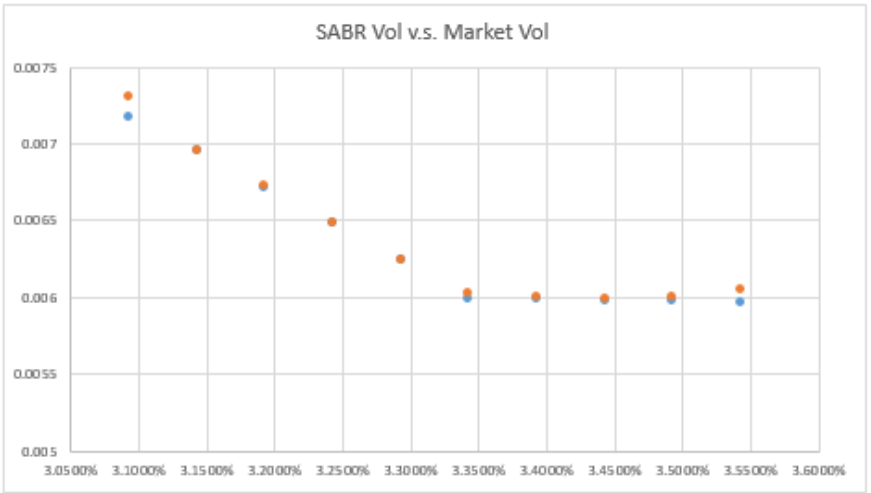


Line Price(kiraCreateValueReport)

User Inputs							Output		
Product Type	Buy	Ntl	Expiry	Tenor	PayOrRec	Strike	ParRate	PV	PV01
SWAP	BUY	10,000.00	3/3/2020	5Y	REC	3.30%	3.17%	5703.35	4.53
SWAP	SELL	20,000.00	2D	2Y	PAY	2.56%	3.37%	-308.96	1.915
FRA	BUY	10,000.00	4M	3M	PAY	2.35%	2.43%	203.059	0.24
FRA	BUY	30,000.00	6M	3M	REC	2.40%	2.97%	4175.44	0.26
SWAPTION	SELL	50,000.00	6/30/2020	10Y	PAY	2.40%	3.25%	-350189	8.03
CAPFLOOR	BUY	40,000.00	5/2/2021	3M	CAP	2.65%	3.25%	5948.07	2.1
SWAPTION	SELL	1,000,000.00	10Y	10Y	PAY	3.43%	3.34%	-43106	6.1

Volatility Smile by SABR

	Strikes	Mkt Vol	SABR Vol
REC	3.0920%	0.00731715	0.00718215
REC	3.1420%	0.006966171	0.00695718
REC	3.1920%	0.006732758	0.00672692
REC	3.2420%	0.006496899	0.0064911
REC	3.2920%	0.006251315	0.00624941
PAY	3.3420%	0.006038365	0.00600092
PAY	3.3920%	0.006012451	0.00599423
PAY	3.4420%	0.00599578	0.00598663
PAY	3.4920%	0.006009308	0.00598241
PAY	3.5420%	0.00606366	0.00597729



Q& A

THANKS!