Swap_Pricing

December 23, 2017

1 Get discount factor for JPY

- input: MoneyMarket (short term interest rate), Swap rate.
- output: discount factors for each tenor listed by MoneMarket and Swap rate.

1.1 Pricing

1.1.1 Swap pricing formula

The value of the exchange between a floot and a fixed side is given by

$$V = \sum_{i=1}^{N} L(t_{i-1}, t_i) \times DF(t_i) \times \delta_i - \sum_{i=1}^{N} SwapRate \times DF(t_i) \times \delta_i,$$

where $L(t_{i-1}, t_i)$ is the floot interest rate between t_{i-1} and t_i, $DF(t_i)$ is a discount factor, δ_i is a day-count-fraction and $SwapRate$ is a Swap rate which means a par rate for a swap trade.

1.1.2 Bootstrap method for getting discount factors

Discount factors as of today can be estimated from a par swap trade which corresponds to $V = 0$ under swap pricing formula. For example, let us consider a swap trade with maturity of 1.5 year. The discount factor for 1.5 year $DF(t_{1.5Y})$ is calculated by solveing the following equation:

$$\sum_{i=1}^{3} L(t_{i-1}, t_i) \times DF(t_i) \times \delta = \sum_{i=1}^{3} SwapRate(1.5Y) \times DF(t_i) \times \delta$$

where a quoted swap rate is used for $SwapRate(1.5Y)$, the day-count-fraction δ is asuumed 6 month and the floot side interest rate is assumed that a following model expressed as

$$L(t_{i-1}, t_i) = \frac{1}{\delta}\Big(\frac{DF(t_{i-1})}{DF(t_i)} - 1\Big).$$

The above equation can be solved by using $DF(t_{0.5Y})$, $DF(t_{1.0Y})$ and the floot interest rate which is defined as above equation. As a result, the discount factor $DF(t_{1.5Y})$ is given by

$$DF(t_{1.5Y}) = \frac{1}{(1 + \delta \times SwapRate(1.5Y))}\Big(DF(t_0) - SwapRate(1.5Y) \times \delta \times \big(DF(t_{0.5Y}) + DF(t_{1.0Y})\big)\Big),$$

where $DF(t_{0.5Y})$ and $DF(t_{1.0Y})$ is calculated by using a quoted LIBOR (the rate of Money Market). The short rate of Money Market means spot rate, where the cashflows is expressed as only two terms. For example, $DF(t_{0.5Y})$ is given by

$$DF(t_{0.5Y}) = \frac{1}{(1 + \delta \times L(0.0Y, 0.5Y))},$$

where $L(0.0Y, 0.5Y)$ is the LIBOR rate between today and 6 month later. Discount factors after $t_{1.5Y}$ can be calculated by the same way as the derivation of $DF(t_{1.5Y})$. This method of getting discount factors gradually is called Bootstrap method.

```
In [11]: import matplotlib.pyplot as plt
         import numpy as np
         import datetime

         class getDF_moneymarket:
         '''    def __init__(self, libor_rate, start_day, end_day):
                 self.libor_rate = libor_rate
                 self.start_day = start_day
                 self.end_day = end_day
                 self.datetime_obj_start = datetime.datetime.strptime(start_day, '%Y/%m/%d')
                 self.datetime_obj_end = datetime.datetime.strptime(end_day, '%Y/%m/%d')
                 self.daycount = (self.datetime_obj_end - self.datetime_obj_start).days / 360
                 self.discount_factor = 0
         '''
             def __init__(self, today, array_ccy):
                     self._start_day = today

             def getDF(self, seq_moneymarket):


In [12]: DF = getDF_moneymarket(0.2, '2017/12/18', '2019/12/30')
         print(DF.discount_factor)
         print(DF.getDF())
         print(DF.discount_factor)

0
[0.7081038552321007, '2017/12/18', '2019/12/30']
[0.7081038552321007, '2017/12/18', '2019/12/30']


In [3]: DF1 = getDF_moneymarket(0.3, '2017/12/18', '2018/3/20')
        DF1.getDF()

Out[3]: 0.9287925696594427

In [76]: import numpy as np
         import csv
         with open('sample_moneymarket.csv', 'r') as csvfile:
```

```
    reader_obj = csv.reader(csvfile)
    # rewritten header_obj by using next method(???)
    header_obj = next(reader_obj)
    mm_list = []
    for row in reader_obj:
        mm_list.append(row)

mm_list

def get_DF(money_market_list):
    list_len = len(money_market_list)
    discount_factor = np.zeros(list_len*2).reshape(list_len, 2)
#    discount_factor = [[] for i in range(list_len)]
#    for i in range(0,list_len):
 #        discount_factor[i][0] = str(discount_factor[i][0])
#    for i in range(0, list_len):
 #        discount_factor[i][0] = money_market_list[i][0]
#    moji = str(discount_factor[0][0])
    return discount_factor
#    return moji

get_DF(mm_list)[0][0]
```

Out[76]: '0'

In [43]:
```
[[] for i in range(5)]
```

Out[43]: [[], [], [], [], []]