Solutions - Practical Lesson 5

Matteo Sani
matteosan1@gmail.com

October 23, 2019

1 Solutions

1.1 Exercises

1.1.1 Exercise 5.1

In the next lesson we're going to build lots of `OvernightIndexSwap` objects, one for each market quote we have. The market quotes will consist of fixed strikes for 1M, 2M, 3M, ..., 12M, 15M, 18M, 2Y, 3Y, ..., 30Y and 40Y swaps.

It would be very boring to write a long list of payment dates for each one of these, plus they'd need to be updated every day. Write a function which given a start date and the number of months, returns a list of dates of **annual** frequency starting from the start date and ending after the specified number of months.

For example

2019-11-10 start date 12 months → 2019-11-10, 2020-11-10 2019-11-10 start date 24 months → 2019-11-10, 2020-11-10, 2021-11-10

Note that if the number of months is not a multiple of 12, the last period should simply be shorter than 12 months. For example

2019-11-10 start date 9 months → 2019-11-10, 2020-08-10 2019-11-10 start date 15 months → 2019-11-10, 2020-11-10, 2021-02-10

Solution:

```
In [1]: from finmarkets import generate_swap_dates
        from datetime import date
        from dateutil.relativedelta import relativedelta

        def generate_swap_dates(start_date, n_months):
            dates = []
            for i in range(0, n_months, 12):
                dates.append(start_date + relativedelta(months=i))
            dates.append(start_date + relativedelta(months=n_months))

            return dates


        assert generate_swap_dates(date(2019, 11, 10), 12) == [date(2019, 11, 10),
                                                               date(2020, 11, 10)]
```

```
        assert generate_swap_dates(date(2019, 11, 10), 24) == [date(2019, 11, 10),
                                                               date(2020, 11, 10),
                                                               date(2021, 11, 10)]

        assert generate_swap_dates(date(2019, 11, 10), 9) == [date(2019, 11, 10),
                                                              date(2020, 8, 10)]
        assert generate_swap_dates(date(2019, 11, 10), 15) == [date(2019, 11, 10),
                                                               date(2020, 11, 10),
                                                               date(2021, 2, 10)]
```

1.1.2 Exercise 5.2

Take the `OvernightIndexSwap` class from the lesson and add a new method called fair_value_strike which takes a discount curve object and returns the fixed rate which would make the OIS have zero NPV.

Hints:

- first take the formulas for the NPV of the fixed leg and the NPV of the floating leg, put one equal to the other and solve for K;
- then implement that in Python.

Solution: As the hint suggested and going back to lesson 5 formulas we have:

$$\text{NPV}_{\text{fix}} = NK \sum_{i=1}^{n} D(d_i) \frac{d_i - d_{i-1}}{360}$$

$$\text{NPV}_{\text{float}} = N \cdot [D(d_0) - D(d_n)]$$

$$K \sum_{i=1}^{n} D(d_i) \frac{d_i - d_{i-1}}{360} = [D(d_0) - D(d_n)]$$

$$K = \frac{[D(d_0) - D(d_n)]}{\sum_{i=1}^{n} D(d_i) \frac{d_i - d_{i-1}}{360}}$$

Now in python:

```
class OverNightIndexSwap:
    ...
    def fair_value_strike(self, discount_curve):
        den = 0
        for i in range(1, len(self.payment_dates)):
            start_date = self.payment_dates[i-1]
            end_date = self.payment_dates[i]
            tau = (end_date - start_date).days / 360
            df = discount_curve.df(end_date)
            den += df * tau
        num = (discount_curve.df(self.payment_dates[0]) -
               discount_curve.df(self.payment_dates[-1]))

        return num/den
```

1.1.3 Exercise 5.3

Take the `OvernightIndexSwap` class, add it to finmarkets.py and try importing it and using it.

Solution: Copy the above function in the `OvernightIndexSwap` class. Then import the finmarket library and check few results

```
In [3]: from datetime import date
        from finmarkets import OvernightIndexSwap, DiscountCurve

        curve = DiscountCurve(date(2019, 1, 1),
                              [date(2019, 1, 1),
                               date(2019, 6, 1),
                               date(2020, 1, 1)],
                              [1.0, 0.98, 0.82])

        ois = OvernightIndexSwap(
            # the notional, one million
            1e6,
            # the list of product dates,
            # i.e. the start date then the payment dates
            [date(2019, 1, 1),
             date(2019, 4, 1),
             date(2019, 7, 1),
             date(2019, 10, 1),
             date(2020, 1, 1)],
            # the fixed rate, 2.5%
            0.025
        )

        ois.fair_value_strike(curve)

Out[3]: 0.1947172768497251
```