

# EIOPA RISK-FREE CURVE JULY-22 RECALCULATION

The risk-free curve is one of the principal inputs into an economic scenario generator. This notebook recalculates the risk-free curve using the parameters that are claimed to be used. The European Insurance and Occupational Pensions Authority (EIOPA) publishes their own yield curve prediction. To do this they use the Smith & Wilson algorithm.

## Summary

The goal of this test is to replicate the EIOPA yield curve. This test will use the methodology that EIOPA claims it is using and the calibration vector that they publish. If the test is passed, the user can be more confident, that EIOPA risk free rate (RFR) curve was generated using the described methodology/calibration and that the process was implemented correctly.

## Table of Contents

1. [Note on Smith & Wilson algorithm](#)
2. [Data requirements](#)
3. [Success criteria](#)
4. [External dependencies](#)
5. [Calibration parameters and calibration vector provided by EIOPA](#)
6. [Smith & Wilson calculation functions](#)
7. [Generation of the risk-free curve](#)
8. [Test 1; Comparison test](#)
9. [Test 1; Success criteria](#)
10. [Test 1; Comparison test](#)
11. [Conclusion](#)

## Note on Smith & Wilson algorithm

To replicate the calculations, this example uses a modified Smith&Wilson implementation (The original implementation is available on [GitHub](#):

- [Python](#)
- [Matlab](#)
- [JavaScript](#)

## Limitations of the implementation

Current implementation only looks at a single currency and without any Volatility Adjustment (VA). The day count convention assumes that each year has the same number of days.

## Data requirements

This script contains the EIOPA risk-free rate publication for July 2022. The publication can be found on the [EIOPA RFR website](#).

The observed maturities `M_Obs` and the calibrated vector `Qb` can be found in the Excel sheet `EIOPA_RFR_20220731_Qb_SW.xlsx`.

For this example, the curve without the volatility adjustment (VA) is used. It can be found in the sheet `SW_Qb_no_VA`. This example is focused on the EUR curve, but this example can be easily modified for any other curve.

The target maturities (`T_Obs`), the additional parameters (`UFR` and `alpha`), and the given curve can be found in the Excel `EIOPA_RFR_20220731_Term_Structures.xlsx`, sheet `RFR_spot_no_VA`.

[Back to the top](#)

## Success criteria

The following success criteria is defined:

- Maximum difference between the calculated curve and the one provided by EIOPA is less than 0.1 bps
- Average difference between the calculated curve and the one provided by EIOPA is less than 0.05 bps

In [1]:

```
test_statistics_max_diff_in_bps = 0.1
test_statistics_average_diff_in_bps = 0.05
```

The success function is called at the end of the test to confirm if the success criteria have been met.

```
In [2]:  
def SuccessTest(TestStatistics, threshold_max, threshold_mean):  
    out1 = False  
    out2 = False  
    if max(TestStatistics) < threshold_max:  
        print("Test passed")  
        out1 = True  
    else:  
        print("Test failed")  
  
    if np.mean(TestStatistics) < threshold_mean:  
        print("Test passed")  
        out2 = True  
    else:  
        print("Test failed")  
    return [out1, out2]
```

[Back to the top](#)

## External dependencies

This implementation uses three well established Python packages widely used in the financial industry. Pandas (<https://pandas.pydata.org/docs/>), Numpy (<https://numpy.org/doc/>), and Matplotlib (<https://matplotlib.org/stable/index.html>)

```
In [3]:  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
%matplotlib notebook
```

[Back to the top](#)

## Calibration parameters and calibration vector provided by EIOPA

```
In [4]:  
# Maturity of observations:  
M_Obs = np.transpose(np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]))  
  
# Ultimate forward rate ufr represents the rate to which the rate curve will converge  
ufr = 0.0345  
  
# Convergence speed parameter alpha controls the speed at which the curve converges  
alpha = 0.126079  
  
# For which maturities do we want the SW algorithm to calculate the rates. In this case  
M_Target = np.transpose(np.arange(1,151))  
  
# Qb calibration vector published by EIOPA for the curve calibration:  
Qb = np.transpose(np.array([  
    7.9853049896658200, -9.3204030221295700, 6.3241509057704200, -1.8348698798880]))
```

[Back to the top](#)

## Smith & Wilson calculation functions

In this step, the independent version of the Smith&Wilson algorithm is implemented. To do this, two functions are taken from the publicly available repository and modified to accept the product of  $Q^*b$  instead of the calibration vector  $b$ .

In [5]:

```
def SWExtrapolate(M_Target, M_Obs, Qb, ufr, alpha):
    # SWEXTRAPOLATE Interpolate or/and extrapolate rates for targeted maturities using a
    # out = SWExtrapolate(M_Target, M_Obs, Qb, ufr, alpha) calculates the rates for maturities
    #
    # Arguments:
    #     M_Target = k x 1 ndarray. Each element represents a bond maturity of interest. Ex. M_Target = [1; 3]
    #     M_Obs = n x 1 ndarray. Observed bond maturities used for calibrating the calibration vector.
    #     Qb = n x 1 ndarray. Calibration vector calculated on observed bonds.
    #     ufr = 1 x 1 floating number. Representing the ultimate forward rate. Ex. ufr = 0.042
    #     alpha = 1 x 1 floating number. Representing the convergence speed parameter alpha
    #
    # Returns:
    #     k x 1 ndarray. Represents the targeted rates for a zero-coupon bond. Each rate is represented by a float.
    #
    # For more information see https://www.eiopa.europa.eu/sites/default/files/risk_free_curve.ipynb

def SWHeart(u, v, alpha):
    # SWHEART Calculate the heart of the Wilson function.
    # H = SWHeart(u, v, alpha) calculates the matrix H (Heart of the Wilson function) for maturities specified by vectors u and v. The formula is
    # taken from the EIOPA technical specifications paragraph 132.
    #
    # Arguments:
    #     u = n_1 x 1 vector of maturities. Ex. u = [1; 3]
    #     v = n_2 x 1 vector of maturities. Ex. v = [1; 2; 3; 5]
    #     alpha = 1 x 1 floating number representing the convergence speed parameter alpha
    #
    # Returns:
    #     n_1 x n_2 matrix representing the Heart of the Wilson function for selected maturities.
    #
    # For more information see https://www.eiopa.europa.eu/sites/default/files/risk_free_curve.ipynb

    u_Mat = np.tile(u, [v.size, 1]).transpose()
    v_Mat = np.tile(v, [u.size, 1])
    return 0.5 * (alpha * (u_Mat + v_Mat) + np.exp(-alpha * (u_Mat + v_Mat))) - alpha

H = SWHeart(M_Target, M_Obs, alpha) # Heart of the Wilson function from paragraph 132
p = np.exp(-np.log(1+ufr)* M_Target) + np.diag(np.exp(-np.log(1+ufr) * M_Target))
return p ** (-1/ M_Target) -1 # Convert obtained prices to rates and return price
```

[Back to the top](#)

## Generation of the risk-free curve

The observed maturities, target maturities, and the model parameters provided by EIOPA are

In [6]:

```
r_Target = SWExtrapolate(M_Target,M_Obs, Qb, ufr, alpha)
r_Target = pd.DataFrame(r_Target,columns=['Recalculated rates'])
```

## **Curve calculated**

## Yield curve

In [7]:

```
r_Target.head()
```

Out[7]:

Recalculated rates	
0	0.008598
1	0.010460
2	0.010891
3	0.011754
4	0.012520

[Back to the top](#)

## Test 1; Comparison test

Comparison of the calculated yield curve with the yield curve provided by EIOPA. The test is passed if the success criteria is reached.

The provided yield curve can be found in file *EIOPA\_RFR\_20220731\_Term\_Structures.xlsx*, sheet *RFR\_spot\_no\_VA*.

In [8]:

```
EUR curve = np.transpose(np.array([0, 0.0860, 0, 0.1046, 0, 0.1089, 0, 0.1175, 0, 0.1252, 0, 0.1316, 0,
```

This implementation looks at two kinds of test statistics. The average deviation and the maximum deviation.

The average deviation is defined as:

$$S_{AVERAGE} = \frac{1}{T} \sum_{t=0}^T |r_{EIOPA}(t) - r_{EST}(t)|$$

The maximum deviation is defined as:

$$S_{MAX} = \max_t |r_{EIOPA}(t) - r_{EST}(t)|$$

Where  $T$  is the maximum maturity available.

The average difference test is successful if:

$$S_{AVVERAGE} < 0.05bps$$

The maximum difference test is successful if:

$$S_{MAX} < 0.1bps$$

```
In [9]: EUR_curve = pd.DataFrame(EUR_curve,columns=['Given rates'])
```

### EIOPA curve provided

Yield curve

```
In [10]: EUR_curve.head()
```

```
Out[10]: Given rates
```

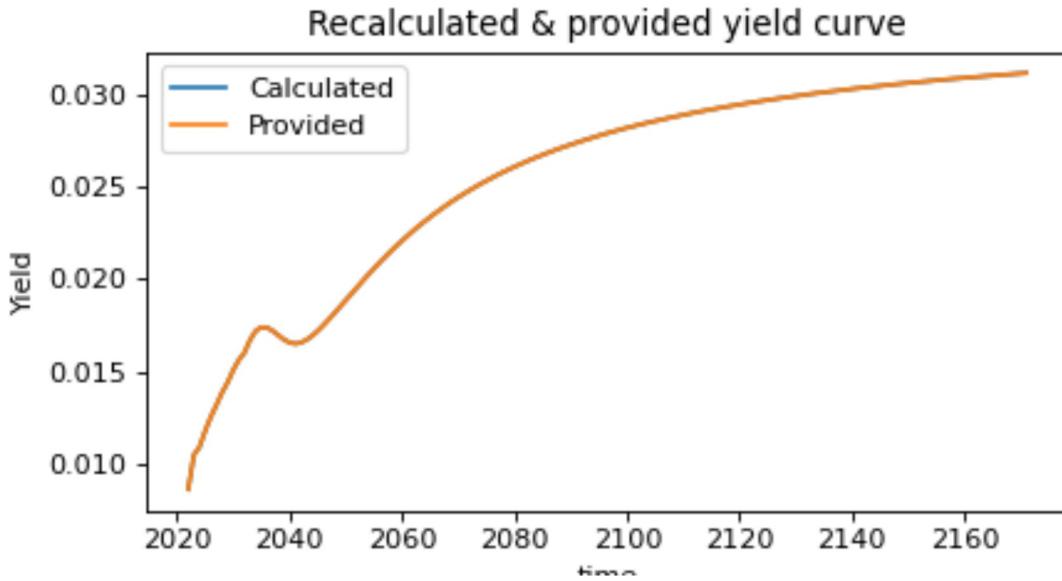
0	0.00860
1	0.01046
2	0.01089
3	0.01175
4	0.01252

```
In [11]: x_data_label = range(2022,2022+r_Target.shape[0],1)
```

In [12]:

```
fig, ax1 = plt.subplots(1,1)
ax1.plot(x_data_label, r_Target.values, color='tab:blue',label="Calculated")
ax1.plot(x_data_label, EUR_curve.values, color='tab:orange',label="Provided")

ax1.set_ylabel("Yield")
ax1.set_title('Recalculated & provided yield curve')
ax1.set_xlabel("time")
ax1.legend()
fig.set_figwidth(6)
fig.set_figheight(3)
plt.show()
```



In [13]:

```
test_statistics_bdp = pd.DataFrame(abs(r_Target.values-EUR_curve.values)*10000, columns=['Abs diff in bps'])
```

### EIOPA curve comparison

Absolute difference in bps

In [14]:

```
test_statistics_bdp.head()
```

Out[14]:

Abs diff in bps

	Abs diff in bps
0	0.015000
1	0.003080
2	0.014340
3	0.043720
4	0.001922

[Back to the top](#)

## Test 1; Success criteria

The successful application of the success criteria marks the completion of the test.

In [15]:

```
result1 = SuccessTest(test_statistics_bdp.values, test_statistics_max_diff_in_bps, tolerance)
```

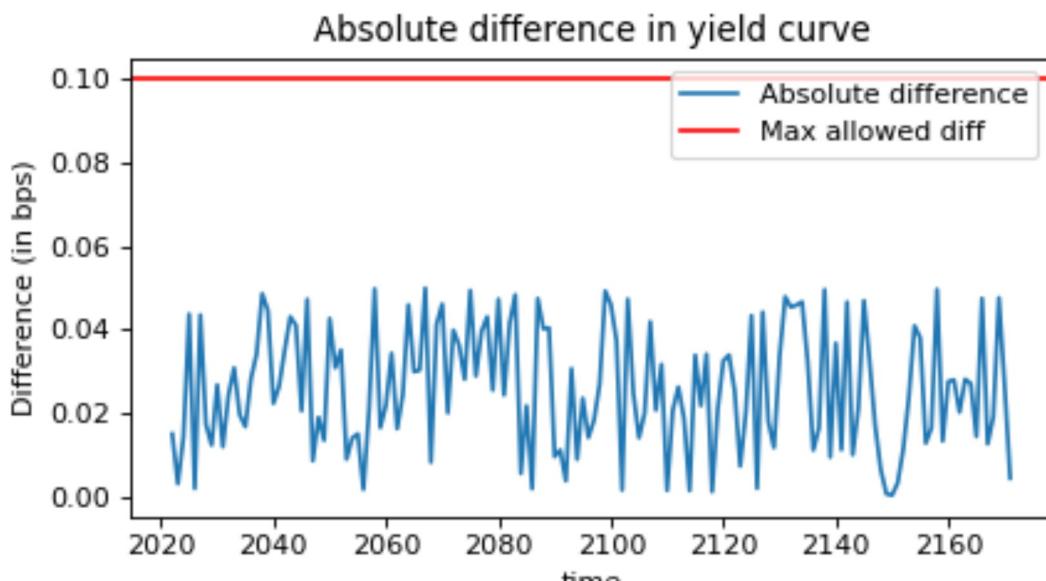
Test passed  
Test passed

In [16]:

```
x_data_label = range(2022,2022+r_Target.shape[0],1)
fig, ax1 = plt.subplots(1,1)
ax1.plot(x_data_label, test_statistics_bdp, label= "Absolute difference")
ax1.axhline(y = test_statistics_max_diff_in_bps, color = 'r', linestyle = ' - ',label='Max allowed diff')

ax1.set_xlabel("time")
ax1.set_ylabel("Difference (in bps)")
ax1.set_title('Absolute difference in yield curve')
ax1.legend()
fig.set_figwidth(6)
fig.set_figheight(3)

plt.show()
```

[Back to the top](#)

## Conclusion

The EIOPA curve generated for July 2022 has passed the success criteria. Based on the preformed tests, it is likely that the curve was generated using the Smith & Wilson algorithm with the calibration vector that was provided in the file *EIOPA\_RFR\_20220731\_Qb\_SW.xlsx* and the

parameters displayed in the file *EIOPA RFR 20220731 Term Structures.xlsx*.

In [17]:

```
pd.DataFrame(data = [result1], columns = ["Mean test", "Max test"], \
              index= ["Provided vs calculated"])
```

Out[17]:

	Mean test	Max test
Provided vs calculated	True	True