

EIOPA RISK-FREE CURVE MARCH-23 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 with/without 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 March 2023. 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_20230331_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_20230331_Term_Structures.xlsx`, sheet `RFR_spot_no_VA` if the test looks at the curve without the Volatility Adjustment and the sheet `RFR_spot_with_VA` if the test looks at the curve with the Volatility Adjustment.

[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 [125...]

```
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 [126...]

```
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 [127...]

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick
%matplotlib notebook
pd.options.display.max_rows = 150
```

Importing data

In [128...]

```
selected_param_file = 'Param_VA.csv'
selected_curves_file = 'Curves_VA.csv'

#selected_param_file = 'Param_no_VA.csv'
#selected_curves_file = 'Curves_no_VA.csv'
```

In [129...]

```
param_raw = pd.read_csv(selected_param_file, sep=',', index_col=0)
```

Parameter input

Parameters sheet

In [130...]

```
param_raw.head()
```

Out[130...]

	Euro_Maturities	Euro_Values	Austria_Maturities	Austria_Values	Belgium_Maturities	Be
Country						
Coupon_freq	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
LLP	20.000000	20.000000	20.000000	20.000000	20.000000	20.000000
Convergence	40.000000	40.000000	40.000000	40.000000	40.000000	40.000000
UFR	3.450000	3.450000	3.450000	3.450000	3.450000	3.450000
alpha	0.113689	0.113689	0.113689	0.113689	0.113689	0.113689

5 rows × 106 columns

The country selected is:

In [131...]

```
country = "United States"
```

In [132...]

```
maturities_country_raw = param_raw.loc[:,country+"_Maturities"].iloc[6:]
param_country_raw = param_raw.loc[:,country + "_Values"].iloc[6:]
extra_param = param_raw.loc[:,country + "_Values"].iloc[:6]
```

Extra parameters

Smith-Wilson calibration parameters

In [133...]

```
extra_param
```

Out[133...]

Country	
Coupon_freq	1.000000
LLP	30.000000
Convergence	40.000000
UFR	3.450000
alpha	0.092029
CRA	0.000000
Name:	United States_Values, dtype: float64

In [134...]

```
relevant_positions = pd.notna(maturities_country_raw.values)
```

In [135...]

```
maturities_country = maturities_country_raw.iloc[relevant_positions]
```

Maturity vector

Vector of maturities used in the calibration

In [136...]

```
maturities_country.head(15)
```

Out[136...]

```
Country
1      1.0
2      2.0
3      3.0
4      4.0
5      5.0
6      6.0
7      7.0
8      8.0
9      9.0
10     10.0
11     11.0
12     12.0
13     13.0
14     14.0
15     15.0
Name: United States_Maturities, dtype: float64
```

In [137...]

```
Qb = param_country_raw.iloc[relevant_positions]
```

Calibration vector

Vector Qb provided as input

In [138...]

```
Qb
```

Out[138...]

```
Country
1      -42.630898
2       16.660765
3        1.049620
4        2.410462
5       -1.033312
6        0.021824
7        0.105421
8        0.011332
9        1.137728
10      -0.817833
11      -0.003711
12      -0.005220
13      -0.004227
14      -0.003246
15       0.325224
16      -0.019159
```

```
17    -0.016250
18    -0.013722
19    -0.013263
20    -0.959279
21     0.039695
22     0.034131
23     0.032225
24     0.026468
25     0.033991
26    -0.006431
27     0.132911
28    -0.398131
29     1.574620
30    -0.895263
Name: United States Values, dtype: float64
```

In [139...]

```
curve_raw = pd.read_csv(selected_curves_file, sep=',', index_col=0)
```

In [140...]

```
curve_country = curve_raw.loc[:, country]
```

[Back to the top](#)

Calibration parameters and calibration vector provided by EIOPA

In [141...]

```
# Maturity of observations:
M_Obs = np.transpose(np.array(maturities_country.values))

# Ultimate forward rate ufr represents the rate to which the rate curve will converge
ufr = extra_param.iloc[3]/100

# Convergence speed parameter alpha controls the speed at which the curve converges
alpha = extra_param.iloc[4]

# For which maturities do we want the SW algorithm to calculate the rates. In this case all
M_Target = np.transpose(np.arange(1, 151))

# Qb calibration vector published by EIOPA for the curve calibration:
Qb = np.transpose(np.array(Qb.values))
```

[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 [142...]

```

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.
    #     M_Obs = n x 1 ndarray. Observed bond maturities used for calibrating the curve.
    #     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 of the
    #
    #
    # Returns:
    #     k x 1 ndarray. Represents the targeted rates for a zero-coupon bond. Each rate is
    #
    # 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 of the
    #
    # Returns:
    #     n_1 x n_2 matrix representing the Heart of the Wilson function for selected
    #
    # 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 used to generate the target curve.

In [143...]

```

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

```

Yield curve calculated

Yield curve calculated using the calibration vector Q_b

In [144...]

```
r_Target.head(15)
```

Out[144...]

Recalculated rates

0	0.052888
1	0.046340
2	0.042410
3	0.040243
4	0.039022
5	0.038253
6	0.037758
7	0.037466
8	0.037327
9	0.037298
10	0.037303
11	0.037318
12	0.037332
13	0.037340
14	0.037338

[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_20230331_Term_Structures.xlsx*, sheet *RFR_spot_no_VA* if the test looks at the curve without the Volatility Adjustment and the sheet *RFR_spot_with_VA* if the test looks at the curve with the Volatility Adjustment.

In [145...]

```
target_curve = np.transpose(np.array(curve_country.values))
```

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:

$$S_{AVERAGE} < 0.05bps$$

The maximum difference test is successful if:

$$S_{MAX} < 0.1bps$$

In [146...]

```
target_curve = pd.DataFrame(target_curve, columns=['Given rates'])
```

EIOPA curve provided

Yield curve provided by EIOPA

In [147...]

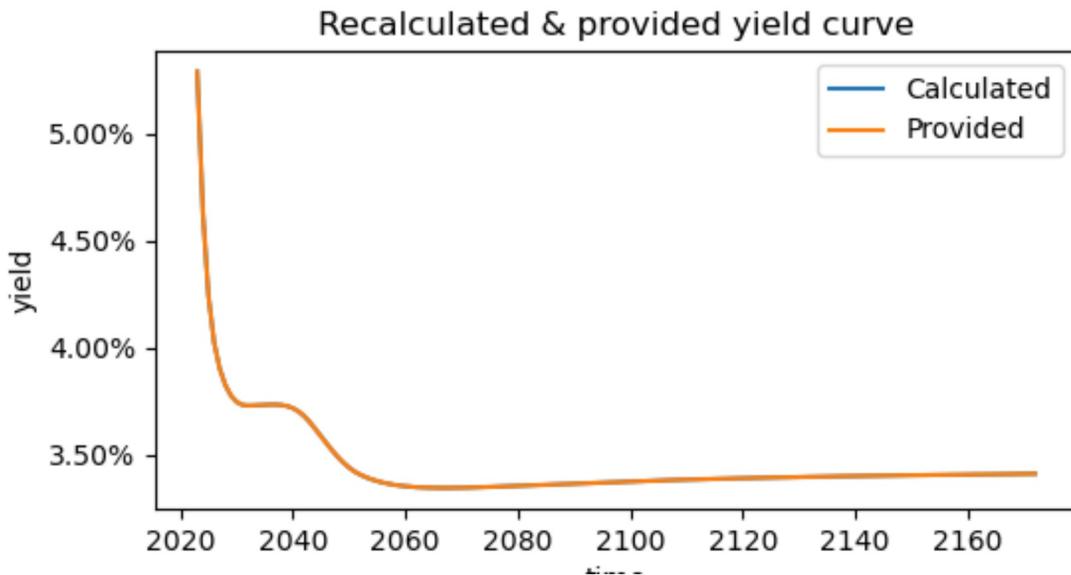
```
target_curve.head()
```

Out[147...]

Given rates	
0	0.05289
1	0.04634
2	0.04241
3	0.04024
4	0.03902

```
In [148...]:  
x_data_label = range(2023,2023+r_Target.shape[0],1)
```

```
In [149...]:  
fig, ax1 = plt.subplots(1,1)  
ax1.plot(x_data_label, r_Target.values*100, color='tab:blue',label="Calculated")  
ax1.plot(x_data_label, target_curve.values*100, color='tab:orange',label="Provided")  
  
ax1.set_ylabel("yield")  
ax1.set_title('Recalculated & provided yield curve')  
ax1.set_xlabel("time")  
ax1.legend()  
ax1.yaxis.set_major_formatter(mtick.PercentFormatter())  
fig.set_figwidth(6)  
fig.set_figheight(3)  
plt.show()
```



```
In [150...]:  
test_statistics_bdp = pd.DataFrame(abs(r_Target.values-target_curve.values)*10000, co
```

EIOPA curve comparison

Absolute difference in bps

```
In [151...]:  
test_statistics_bdp.head()
```

```
Out[151...]:  
Abs diff in bps
```

0	0.020001
1	0.000048
2	0.004152
3	0.026522

Abs diff in bps[Back to the top](#)

Test 1; Success criteria

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

In [152...]

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

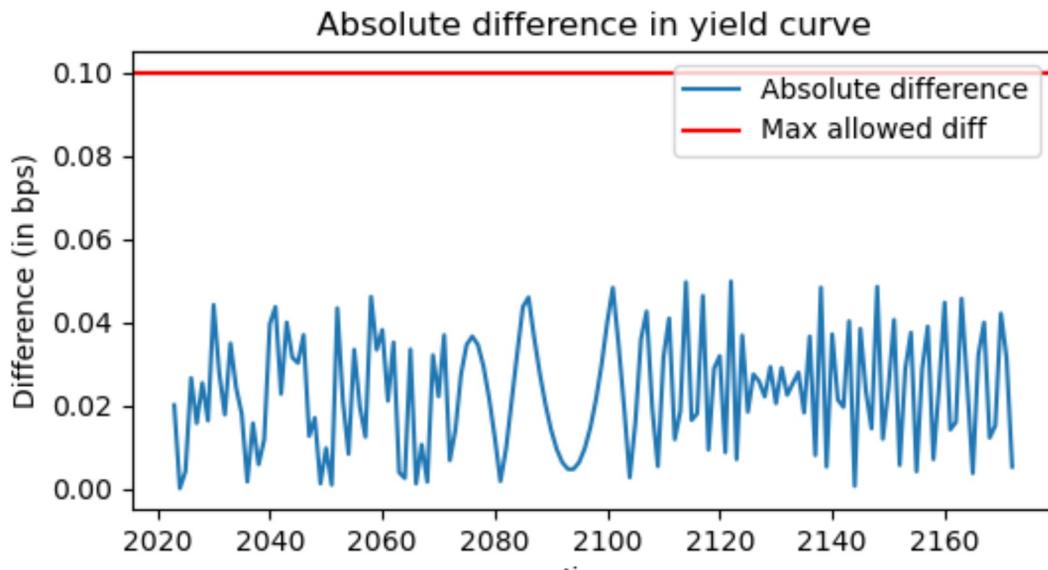
Test passed
Test passed

In [153...]

```
x_data_label = range(2023,2023+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

This test checks the success criteria on the EIOPA curve generated for March 2023. If the tests are passed, it is likely that the curve was generated using the Smith & Wilson algorithm with the

In [154...]

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

Out[154...]

	Mean test	Max test
Provided vs calculated	True	True

Final yield curve

Full yield curve provided by EIOPA in %

In [155...]

```
(curve_country*100).head(150)
```

Out[155...]

Country	
1	5.289
2	4.634
3	4.241
4	4.024
5	3.902
6	3.825
7	3.776
8	3.747
9	3.733
10	3.730
11	3.730
12	3.732
13	3.733
14	3.734
15	3.734
16	3.732
17	3.727
18	3.718
19	3.702
20	3.681
21	3.653
22	3.620
23	3.586
24	3.552
25	3.520
26	3.490
27	3.463
28	3.440
29	3.421
30	3.407
31	3.395
32	3.385
33	3.377

34	3.370
35	3.364
36	3.359
37	3.356
38	3.353
39	3.350
40	3.348
41	3.347
42	3.346
43	3.345
44	3.345
45	3.345
46	3.345
47	3.345
48	3.346
49	3.346
50	3.347
51	3.348
52	3.349
53	3.350
54	3.351
55	3.352
56	3.353
57	3.354
58	3.355
59	3.356
60	3.357
61	3.358
62	3.359
63	3.360
64	3.362
65	3.363
66	3.364
67	3.365
68	3.366
69	3.367
70	3.368
71	3.369
72	3.370
73	3.371
74	3.372
75	3.373
76	3.374
77	3.375
78	3.376
79	3.376
80	3.377
81	3.378
82	3.379
83	3.380
84	3.381
85	3.381
86	3.382
87	3.383
88	3.384
89	3.384
90	3.385
91	3.386
92	3.386

93	3.387
94	3.388
95	3.388
96	3.389
97	3.390
98	3.390
99	3.391
100	3.391
101	3.392
102	3.393
103	3.393
104	3.394
105	3.394
106	3.395
107	3.395
108	3.396
109	3.396
110	3.397
111	3.397
112	3.398
113	3.398
114	3.399
115	3.399
116	3.400
117	3.400
118	3.400
119	3.401
120	3.401
121	3.402
122	3.402
123	3.402
124	3.403
125	3.403
126	3.404
127	3.404
128	3.404
129	3.405
130	3.405
131	3.405
132	3.406
133	3.406
134	3.406
135	3.407
136	3.407
137	3.407
138	3.408
139	3.408
140	3.408
141	3.408
142	3.409
143	3.409
144	3.409
145	3.410
146	3.410
147	3.410
148	3.410
149	3.411
150	3.411

Name: United States, dtvpe: float64

