EIOPA RISK-FREE CURVE JANUARY-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

Note on Smith & Wilson algorithm

To replicate the calculations, this example uses a modified Smith&Wilson implementation (The original implementation is availible 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 January 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_20230131_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_20230131_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 [373…

```
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 [374...

```
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 [375...

```
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 [376...

```
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 [377...

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

Parameter input

Parameters sheet

In [378...
```
param_raw.head()
```

Out[378...

Country	Euro_Maturities	Euro_Values	Austria_Maturities	Austria_Values	Belgium_Maturities	Be
Coupon_freq	1.000000	1.000000	1.000000	1.000000	1.000000	
LLP	20.000000	20.000000	20.000000	20.000000	20.000000	
Convergence	40.000000	40.000000	40.000000	40.000000	40.000000	
UFR	3.450000	3.450000	3.450000	3.450000	3.450000	
alpha	0.116683	0.116683	0.116683	0.116683	0.116683	

5 rows × 106 columns

The country selected is:

In [379...
```
country = "United States"
```

In [380...
```
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 [381...
```
extra_param
```

Out[381...
```
Country
Coupon_freq     1.000000
LLP            30.000000
Convergence    40.000000
UFR             3.450000
alpha           0.091509
CRA             0.000000
Name: United States_Values, dtype: float64
```

In [382...
```
relevant_positions = pd.notna(maturities_country_raw.values)
```

In [383...
```
maturities_country = maturities_country_raw.iloc[relevant_positions]
```

Maturity vector

Vector of maturities used in the calibration

In [384...
```
maturities_country.head(15)
```

Out[384...
```
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 [385...
```
Qb = param_country_raw.iloc[relevant_positions]
```

Calibration vector

Vector Qb provided as input

In [386...
```
Qb
```

Out[386...
```
Country
1     -33.652161
2       2.665230
3      10.467747
4      -1.315085
5       0.659676
6      -0.008444
7      -0.505022
8       0.021452
9       1.045533
10     -0.450041
11     -0.010016
12     -0.010142
13     -0.008845
14     -0.007501
15      0.250207
16     -0.018710
```

```
17     -0.016043
18     -0.013571
19     -0.013071
20     -0.946242
NaN     0.038359
NaN     0.033081
NaN     0.031240
NaN     0.025748
NaN     0.032874
NaN    -0.005568
NaN     0.126840
NaN    -0.377896
NaN     1.497002
NaN    -0.838838
Name: United States Values, dtype: float64
```

In [387…

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

In [388…

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

Back to the top

Calibration parameters and calibration vector provided by EIOPA

In [389…

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

# Ultimate froward 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 t
alpha = extra_param.iloc[4]

# For which maturities do we want the SW algorithm to calculate the rates. In this ca
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 [390…

```
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 matu
#
# Arguments:
#    M_Target = k x 1 ndarray. Each element represents a bond maturity of interest. E
#    M_Obs =    n x 1 ndarray. Observed bond maturities used for calibrating the cal
#    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 o
#
#
# Returns:
#    k x 1 ndarray. Represents the targeted rates for a zero-coupon bond. Each rate E
#
# For more information see https://www.eiopa.europa.eu/sites/default/files/risk_free_

    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 o
    #
    # 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_f

        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)) - al

    H = SWHeart(M_Target, M_Obs, alpha) # Heart of the Wilson function from paragraph
    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 [391…

```
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 Qb

In [392…
```
r_Target.head(15)
```

Out[392…

	Recalculated rates
0	0.053868
1	0.047612
2	0.043044
3	0.040616
4	0.039231
5	0.038393
6	0.037864
7	0.037525
8	0.037340
9	0.037280
10	0.037282
11	0.037310
12	0.037348
13	0.037383
14	0.037406

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_20230131_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 [393…
```
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 [394...]
```
target_curve = pd.DataFrame(target_curve,columns=['Given rates'])
```

EIOPA curve provided

Yield curve provided by EIOPA

In [395...]
```
target_curve.head()
```

Out[395...]

	Given rates
0	0.05387
1	0.04761
2	0.04304
3	0.04062
4	0.03923

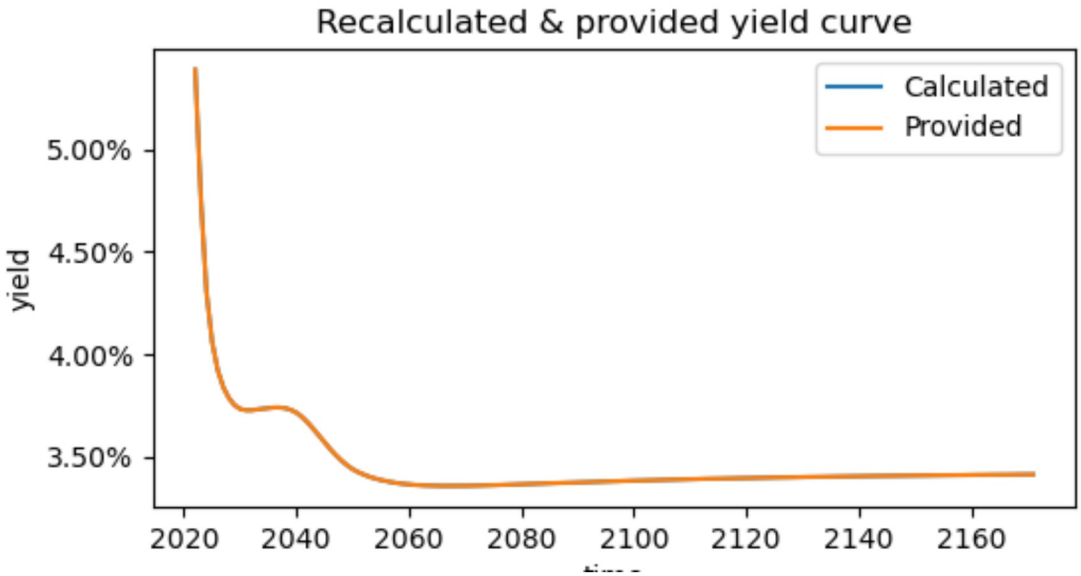
In [396...

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

In [397...

```
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 [398...

```
test_statistics_bdp = pd.DataFrame(abs(r_Target.values-target_curve.values)*10000, c
```

EIOPA curve comparison

Absolute difference in bps

In [399...

```
test_statistics_bdp.head()
```

Out[399...

	Abs diff in bps
0	0.019999
1	0.015614
2	0.043685
3	0.039736

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 [400…

```
result1 = SuccessTest(test_statistics_bdp.values, test_statistics_max_diff_in_bps, te
```

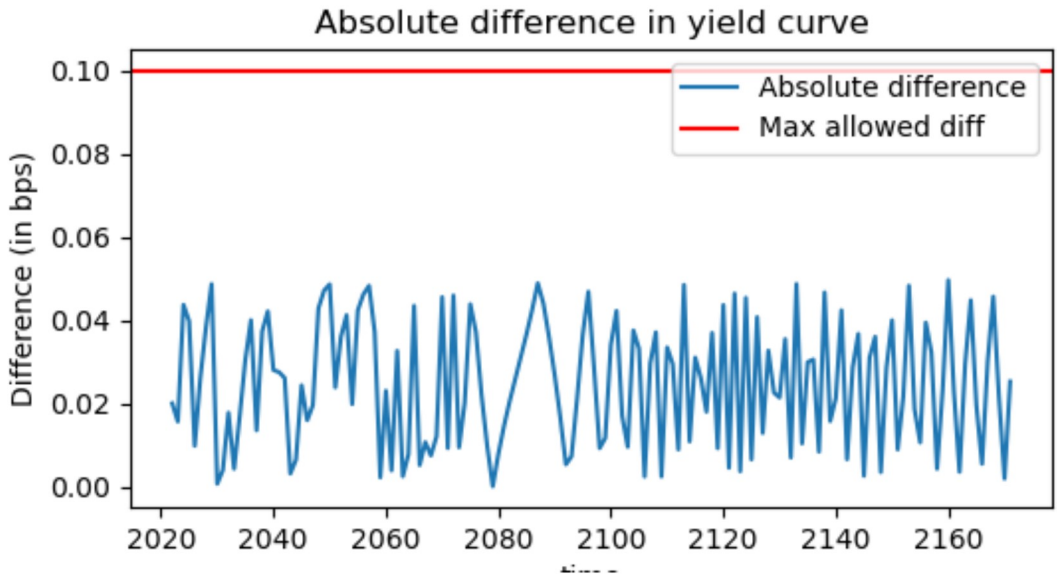
```
Test passed
Test passed
```

In [401…

```
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='

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 January 2023. If the tests are passed, it is likely that the curve was generated using the Smith & Wilson algorithm with the

In [402...
```
pd.DataFrame(data = [result1], columns = ["Mean test","Max test"],  \
             index= ["Provided vs calculated"])
```

Out[402...

	Mean test	Max test
Provided vs calculated	True	True

Final yield curve

Full yield curve provided by EIOPA in %

In [403...
```
(curve_country*100).head(150)
```

Out[403...
```
Country
1      5.387
2      4.761
3      4.304
4      4.062
5      3.923
6      3.839
7      3.786
8      3.752
9      3.734
10     3.728
11     3.728
12     3.731
13     3.735
14     3.738
15     3.741
16     3.741
17     3.738
18     3.730
19     3.716
20     3.695
21     3.668
22     3.637
23     3.604
24     3.571
25     3.539
26     3.509
27     3.483
28     3.460
29     3.440
30     3.426
31     3.414
32     3.403
33     3.395
```

34	3.388
35	3.382
36	3.377
37	3.372
38	3.369
39	3.366
40	3.364
41	3.362
42	3.361
43	3.360
44	3.359
45	3.359
46	3.359
47	3.359
48	3.359
49	3.359
50	3.360
51	3.360
52	3.361
53	3.362
54	3.363
55	3.363
56	3.364
57	3.365
58	3.366
59	3.367
60	3.368
61	3.369
62	3.370
63	3.371
64	3.372
65	3.373
66	3.374
67	3.374
68	3.375
69	3.376
70	3.377
71	3.378
72	3.379
73	3.380
74	3.381
75	3.381
76	3.382
77	3.383
78	3.384
79	3.385
80	3.385
81	3.386
82	3.387
83	3.388
84	3.388
85	3.389
86	3.390
87	3.390
88	3.391
89	3.392
90	3.392
91	3.393
92	3.394

```
93     3.394
94     3.395
95     3.395
96     3.396
97     3.396
98     3.397
99     3.397
100    3.398
101    3.398
102    3.399
103    3.399
104    3.400
105    3.400
106    3.401
107    3.401
108    3.402
109    3.402
110    3.403
111    3.403
112    3.403
113    3.404
114    3.404
115    3.405
116    3.405
117    3.405
118    3.406
119    3.406
120    3.407
121    3.407
122    3.407
123    3.408
124    3.408
125    3.408
126    3.409
127    3.409
128    3.409
129    3.410
130    3.410
131    3.410
132    3.411
133    3.411
134    3.411
135    3.411
136    3.412
137    3.412
138    3.412
139    3.413
140    3.413
141    3.413
142    3.413
143    3.414
144    3.414
145    3.414
146    3.414
147    3.415
148    3.415
149    3.415
150    3.415
Name: United States, dtype: float64
```