

EIOPA RISK-FREE CURVE JUNE-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 June 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_20230630_Qb_SW.xlsx`.

The target maturities (`T_Obs`), the additional parameters (`UFR` and `alpha`), and the given curve can be found in the Excel `EIOPA_RFR_20230630_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 [157...]

```
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 [158...]

```
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 [159...]

```
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 [160...]

```
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 [161...]

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

Parameter input

Parameters sheet

In [162...]

```
param_raw.head()
```

Out[162...]

	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	0.111987	0.111987	0.111987	0.111987	0.111987	0.111987

5 rows × 106 columns

The country selected is:

In [163...]

```
country = "Turkey"
```

In [164...]

```
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 [165...]

```
extra_param
```

Out[165...]

Country	
Coupon_freq	0.000000
LLP	9.000000
Convergence	51.000000
UFR	5.500000
alpha	0.165319
CRA	10.000000
Name:	Turkey_Values, dtype: float64

In [166...]

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

In [167...]

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

Maturity vector

Vector of maturities used in the calibration

In [168...]

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

Out[168...]

```
Country
1    1.0
2    2.0
3    3.0
4    4.0
5    5.0
6    6.0
7    8.0
8    9.0
Name: Turkey_Maturities, dtype: float64
```

In [169...]

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

Calibration vector

Vector **Qb** provided as input

In [170...]

```
Qb
```

Out[170...]

```
Country
1    2.763976
2    1.871045
3   -0.692405
4   -0.523172
5   -0.844859
6   -0.179319
7   -0.136517
8   -0.156069
Name: Turkey_Values, dtype: float64
```

In [171...]

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

In [172...]

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

[Back to the top](#)

Calibration parameters and calibration vector provided

bv, EIOPA

In [173...]

```
# 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 to the target  
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 [174...]

```

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
    #
    # 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 used to generate the target curve.

In [175...]

```

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 [176...]

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

Out[176...]

Recalculated rates

0	0.128955
1	0.137800
2	0.148035
3	0.157283
4	0.164767
5	0.170364
6	0.174457
7	0.177373
8	0.179220
9	0.180106
10	0.180217
11	0.179713
12	0.178709
13	0.177292
14	0.175536

[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_20230630_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 [177...]

```
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 [178...]

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

EIOPA curve provided

Yield curve provided by EIOPA

In [179...]

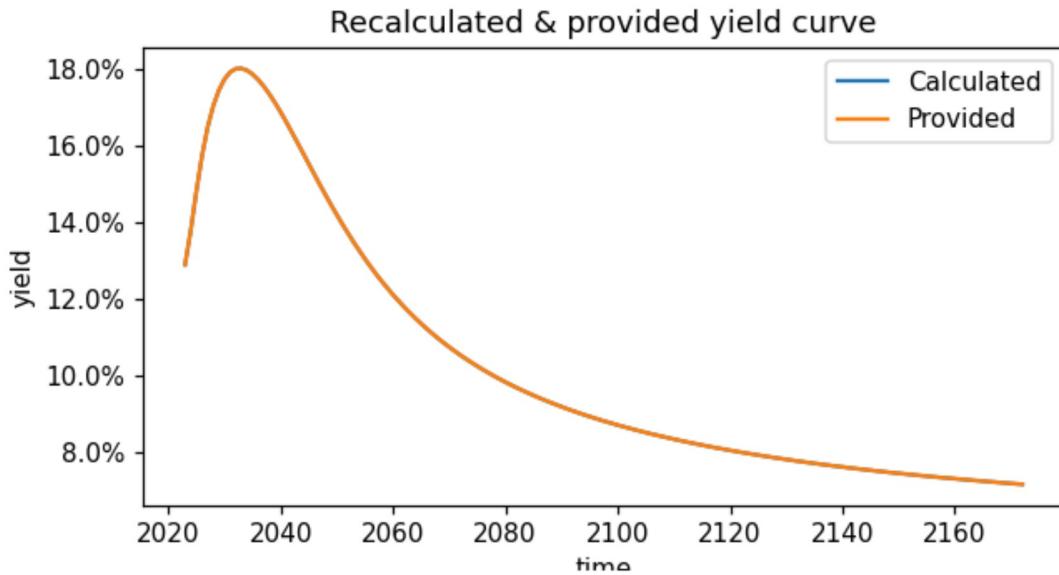
```
target_curve.head()
```

Out[179...]

	Given rates
0	0.12896
1	0.13780
2	0.14803
3	0.15728
4	0.16477

```
In [180...]:  
x_data_label = range(2023,2023+r_Target.shape[0],1)
```

```
In [181...]:  
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 [182...]:  
test_statistics_bdp = pd.DataFrame(abs(r_Target.values-target_curve.values)*10000, co
```

EIOPA curve comparison

Absolute difference in bps

```
In [183...]:  
test_statistics_bdp.head()
```

```
Out[183...]:  
Abs diff in bps
```

	Abs diff in bps
0	0.047840
1	0.004590
2	0.048918
3	0.028604

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.

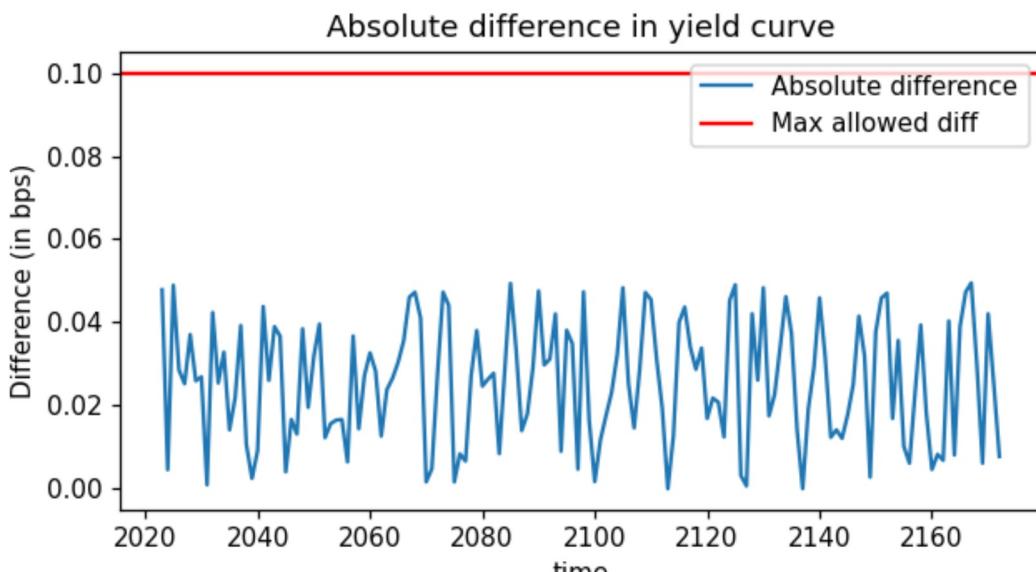
In [184...]

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

```
Test passed  
Test passed
```

In [185...]

```
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 June 2023. If the tests are passed, 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_20230630_Qb_SW.xlsx* and the parameters displayed in the file *EIOPA_RFR_20230630_Term_Structures.xlsx*.

In [186...]

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

Out[186...]

	Mean test	Max test
--	-----------	----------

Provided vs calculated	True	True
------------------------	------	------

Final yield curve

Full yield curve provided by EIOPA in %

In [187...]

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

Out[187...]

Country	
1	12.896
2	13.780
3	14.803
4	15.728
5	16.477
6	17.036
7	17.446
8	17.737
9	17.922
10	18.011
11	18.022
12	17.971
13	17.871
14	17.729
15	17.554
16	17.350
17	17.124
18	16.880
19	16.623
20	16.355
21	16.082
22	15.805
23	15.527
24	15.251
25	14.978
26	14.710
27	14.449
28	14.194

29	13.948
30	13.709
31	13.479
32	13.258
33	13.045
34	12.841
35	12.645
36	12.458
37	12.278
38	12.106
39	11.942
40	11.784
41	11.633
42	11.489
43	11.350
44	11.218
45	11.090
46	10.968
47	10.851
48	10.739
49	10.631
50	10.527
51	10.427
52	10.331
53	10.239
54	10.150
55	10.064
56	9.981
57	9.901
58	9.824
59	9.750
60	9.678
61	9.608
62	9.541
63	9.475
64	9.412
65	9.351
66	9.292
67	9.234
68	9.178
69	9.124
70	9.072
71	9.021
72	8.971
73	8.923
74	8.876
75	8.830
76	8.785
77	8.742
78	8.700
79	8.659
80	8.619
81	8.580
82	8.542
83	8.505
84	8.468
85	8.433
86	8.398
87	8.365

88	8.332
89	8.299
90	8.268
91	8.237
92	8.207
93	8.177
94	8.149
95	8.120
96	8.093
97	8.066
98	8.039
99	8.013
100	7.988
101	7.963
102	7.938
103	7.915
104	7.891
105	7.868
106	7.845
107	7.823
108	7.802
109	7.780
110	7.759
111	7.739
112	7.718
113	7.699
114	7.679
115	7.660
116	7.641
117	7.623
118	7.605
119	7.587
120	7.569
121	7.552
122	7.535
123	7.518
124	7.502
125	7.486
126	7.470
127	7.454
128	7.439
129	7.423
130	7.409
131	7.394
132	7.379
133	7.365
134	7.351
135	7.337
136	7.324
137	7.310
138	7.297
139	7.284
140	7.271
141	7.258
142	7.246
143	7.234
144	7.221
145	7.210
146	7.198

147	7.186
148	7.175
149	7.163
150	7.152
151	7.141
152	7.130
153	7.119
154	7.108
155	7.096
156	7.084
157	7.072
158	7.059
159	7.046
160	7.033
161	7.020
162	7.006
163	6.992
164	6.977
165	6.962
166	6.946
167	6.930
168	6.913
169	6.896
170	6.878
171	6.860
172	6.841
173	6.822
174	6.802
175	6.782
176	6.761
177	6.740
178	6.718
179	6.696
180	6.673
181	6.650
182	6.626
183	6.602
184	6.577
185	6.552
186	6.526
187	6.500
188	6.473
189	6.446
190	6.418
191	6.390
192	6.361
193	6.332
194	6.302
195	6.272
196	6.241
197	6.210
198	6.178
199	6.146
200	6.113
201	6.080
202	6.046
203	6.012
204	5.977
205	5.942
206	5.906
207	5.869
208	5.832
209	5.794
210	5.756
211	5.717
212	5.678
213	5.638
214	5.598
215	5.557
216	5.516
217	5.474
218	5.432
219	5.389
220	5.346
221	5.302
222	5.258
223	5.213
224	5.168
225	5.122
226	5.076
227	5.029
228	5.982
229	5.934
230	5.885
231	5.836
232	5.786
233	5.736
234	5.685
235	5.634
236	5.583
237	5.531
238	5.479
239	5.427
240	5.374
241	5.321
242	5.268
243	5.214
244	5.160
245	5.106
246	5.051
247	4.996
248	4.941
249	4.885
250	4.829
251	4.772
252	4.715
253	4.657
254	4.600
255	4.541
256	4.483
257	4.424
258	4.365
259	4.306
260	4.246
261	4.187
262	4.127
263	4.067
264	4.007
265	3.947
266	3.886
267	3.826
268	3.765
269	3.705
270	3.644
271	3.584
272	3.523
273	3.462
274	3.402
275	3.341
276	3.279
277	3.219
278	3.157
279	3.096
280	3.034
281	2.972
282	2.909
283	2.847
284	2.784
285	2.722
286	2.659
287	2.606
288	2.543
289	2.480
290	2.417
291	2.354
292	2.291
293	2.227
294	2.164
295	2.101
296	2.037
297	1.974
298	1.910
299	1.847
300	1.783
301	1.719
302	1.655
303	1.591
304	1.526
305	1.462
306	1.397
307	1.333
308	1.268
309	1.203
310	1.138
311	1.073
312	1.008
313	9.43
314	8.76
315	8.10
316	7.43
317	6.77
318	6.10
319	5.43
320	4.77
321	4.10
322	3.43
323	2.77
324	2.10
325	1.43
326	0.77
327	0.10
328	-0.53
329	-1.19
330	-1.86
331	-2.53
332	-3.20
333	-3.87
334	-4.54
335	-5.21
336	-5.88
337	-6.55
338	-7.22
339	-7.89
340	-8.56
341	-9.23
342	-9.90
343	-10.57
344	-11.24
345	-11.91
346	-12.58
347	-13.25
348	-13.92
349	-14.59
350	-15.26
351	-15.93
352	-16.60
353	-17.27
354	-17.94
355	-18.61
356	-19.28
357	-19.95
358	-20.62
359	-21.29
360	-21.96
361	-22.63
362	-23.30
363	-23.97
364	-24.64
365	-25.31
366	-25.98
367	-26.65
368	-27.32
369	-27.99
370	-28.66
371	-29.33
372	-30.00
373	-30.67
374	-31.34
375	-32.01
376	-32.68
377	-33.35
378	-34.02
379	-34.69
380	-35.36
381	-36.03
382	-36.70
383	-37.37
384	-38.04
385	-38.71
386	-39.38
387	-40.05
388	-40.72
389	-41.39
390	-42.06
391	-42.73
392	-43.40
393	-44.07
394	-44.74
395	-45.41
396	-46.08
397	-46.75
398	-47.42
399	-48.09
400	-48.76
401	-49.43
402	-50.10
403	-50.77
404	-51.44
405	-52.11
406	-52.78
407	-53.45
408	-54.12
409	-54.79
410	-55.46
411	-56.13
412	-56.80
413	-57.47
414	-58.14
415	-58.81
416	-59.48
417	-60.15
418	-60.82
419	-61.49
420	-62.16
421	-62.83
422	-63.50
423	-64.17
424	-64.84
425	-65.51
426	-66.18
427	-66.85
428	-67.52
429	-68.19
430	-68.86
431	-69.53
432	-70.20
433	-70.87
434	-71.54
435	-72.21
436	-72.88
437	-73.55
438	-74.22
439	-74.89
440	-75.56
441	-76.23
442	-76.90
443	-77.57
444	-78.24
445	-78.91
446	-79.58
447	-80.25
448	-80.92
449	-81.59
450	-82.26
451	-82.93
452	-83.60
453	-84.27
454	-84.94
455	-85.61
456	-86.28
457	-86.95
458	-87.62
459	-88.29
460	-88.96
461	-89.63
462	-90.30
463	-90.97
464	-91.64
465	-92.31
466	-92.98
467	-93.65
468	-94.32
469	-94.99
470	-95.66
471	-96.33
472	-97.00
473	-97.67
474	-98.34
475	-99.01
476	-99.68
477	-100.35
478	-101.02
479	-101.69
480	-102.36
481	-103.03
482	-103.70
483	-104.37
484	-105.04
485	-105.71
486	-106.38
487	-107.05
488	-107.72
489	-108.39
490	-109.06
491	-109.73
492	-110.40
493	-111.07
494	-111.74
495	-112.41
496	-113.08
497	-113.75
498	-114.42
499	-115.09
500	-115.76
501	-116.43
502	-117.10
503	-117.77
504	-118.44
505	-119.11
506	-119.78
507	-120.45
508	-121.12
509	-121.79
510	-122.46
511	-123.13
512	-123.80
513	-124.47
514	-125.14
515	-125.81
516	-126.48
517	-127.15
518	-127.82
519	-128.49
520	-129.16
521	-129.83
522	-130.50
523	-131.17
524	-131.84
525	-132.51
526	-133.18
527	-133.85
528	-134.52
529	-135.19
530	-135.86
531	-136.53
532	-137.20
533	-137.87
534	-138.54
535	-139.21
536	-139.88
537	-140.55
538	-141.22
539	-141.89
540	-142.56
541	-143.23
542	-143.90
543	-144.57
544	-145.24
545	-145.91
546	-146.58
547	-147.25
548	-147.92
549	-148.59
550	-149.26
551	-150.00
552	-150.73
553	-151.47
554	-152.20
555	-152.93
556	-153.66
557	-154.39
558	-155.12
559	-155.85
560	-156.58
561	-157.31
562	-158.04
563	-158.77
564	-159.50
565	-160.23
566	-160.96
567	-161.69
568	-162.42
569	-163.15
570	-163.88
571	-164.61
572	-165.34
573	-166.07
574	-166.80
575	-167.53
576	-168.26
577	-168.99
578	-169.72
579	-170.45
580	-171.18
581	-171.91
582	-172.64
583	-173.37
584	-174.10
585	-174.83
586	-175.56
587	-176.29
588	-177.02
589	-177.75
590	-178.48
591	-179.21
592	-180.00
593	-180.73
594	-181.47
595	-182.20
596	-182.93
597	-183.66
598	-184.39
599	-185.12
600	-185.85
601	-186.58
602	-187.31
603	-188.04
604	-188.77
605	-189.50
606	-190.23
607	-190.96
608	-191.69
609	-192.42
610	-193.15
611	-193.88
612	-194.61
613	-195.34
614	-196.07
615	-196.80
616	-197.53
617	-198.26
618	-198.99
619	-199.72
620	-200.45
621	-201.18
622	-201.91
623	-202.64
624	-203.37
625	-204.10
626	-204.83
627	-205.56
628	-206.29
629	-207.02
630	-207.75
631	-208.48
632	-209.21
633	-210.00
634	-210.73
635	-211.47
636	-212.20
637	-212.93
638	-213.66
639	-214.39
640	-215.12
641	-215.85
642	-216.58
643	-217.31
644	-218.04
645	-218.77
646	-219.50
647	-220.23
648	-220.96