

# EIOPA RISK-FREE CURVE APRIL-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 April 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_20230430_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_20230430_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 [218...]

```
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 [219...]

```
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 [220...]

```
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 [221...]

```
#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 [222...]

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

## Parameter input

Parameters sheet

In [223...]

```
param_raw.head()
```

Out[223...]

	Euro_Maturities	Euro_Values	Austria_Maturities	Austria_Values	Belgium_Maturities	Be
Country						
<b>Coupon_freq</b>	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
<b>LLP</b>	20.000000	20.000000	20.000000	20.000000	20.000000	20.000000
<b>Convergence</b>	40.000000	40.000000	40.000000	40.000000	40.000000	40.000000
<b>UFR</b>	3.450000	3.450000	3.450000	3.450000	3.450000	3.450000
<b>alpha</b>	0.115699	0.115699	0.115699	0.115699	0.115699	0.115699

5 rows × 106 columns

The country selected is:

In [224...]

```
country = "Slovenia"
```

In [225...]

```
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 [226...]

```
extra_param
```

Out[226...]

Country	
Coupon_freq	1.000000
LLP	20.000000
Convergence	40.000000
UFR	3.450000
alpha	0.115699
CRA	10.000000

Name: Slovenia\_Values, dtype: float64

In [227...]

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

In [228...]

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

## Maturity vector

Vector of maturities used in the calibration

In [229...]

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

Out[229...]

```
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: Slovenia_Maturities, dtype: float64
```

In [230...]

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

## Calibration vector

Vector **Qb** provided as input

In [231...]

```
Qb
```

Out[231...]

```
Country
1      -8.096518
2       0.461906
3      1.746009
4      0.966140
5      -0.432827
6      -0.356785
7       0.653077
8      -0.350523
9      -0.365608
10     2.512354
11     -3.619813
12     2.495825
13     -0.024658
14     -0.023836
15     -1.577265
16      0.021297
```

```
17    0.020587
18    0.019900
19    0.019236
20    0.689163
Name: Slovenia Values, dtype: float64
```

```
In [232...]: curve_raw = pd.read_csv(selected_curves_file, sep=',', index_col=0)
```

```
In [233...]: curve_country = curve_raw.loc[:,country]
```

[Back to the top](#)

## Calibration parameters and calibration vector provided by EIOPA

```
In [234...]: # 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, we want rates for all maturities
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 [235...]

```

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; 2; 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 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 [236...]

```

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 [237...]

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

Out[237...]

#### Recalculated rates

0	0.036730
1	0.033619
2	0.031276
3	0.029977
4	0.029317
5	0.028934
6	0.028716
7	0.028646
8	0.028661
9	0.028752
10	0.028899
11	0.028964
12	0.029032
13	0.029051
14	0.028947

---

[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\_20230430\_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 [238...]

```
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 [239...]

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

### EIOPA curve provided

Yield curve provided by EIOPA

In [240...]

```
target_curve.head()
```

Out[240...]

	Given rates
0	0.03673
1	0.03362
2	0.03128
3	0.02998
4	0.02932

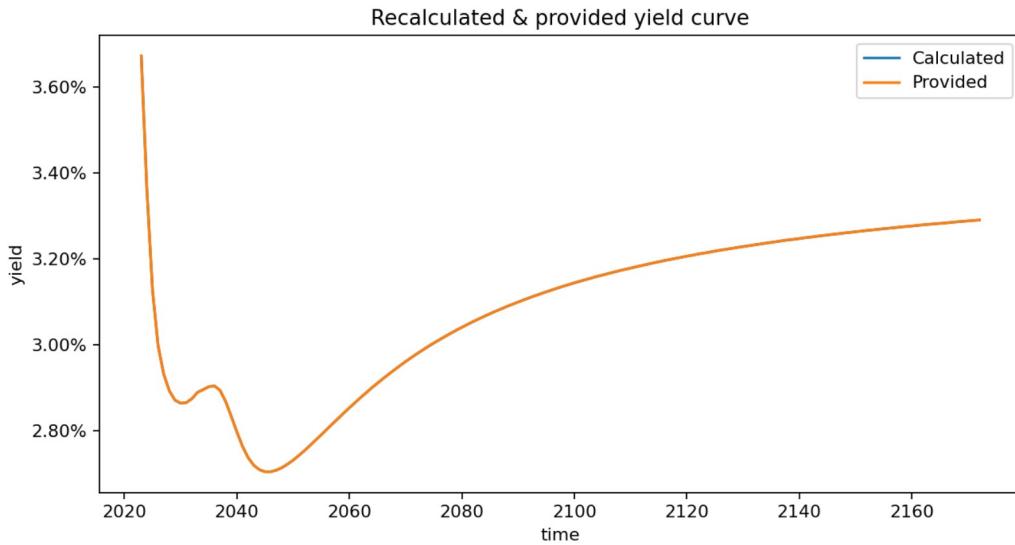
In [241...]

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

In [242...]

```
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 [243...]

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

### EIOPA curve comparison

Absolute difference in bps

In [244...]

```
test_statistics_bdp.head()
```

Out[244...]

	Abs diff in bps
0	0.001081
1	0.014663
2	0.041840
3	0.030074

**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.

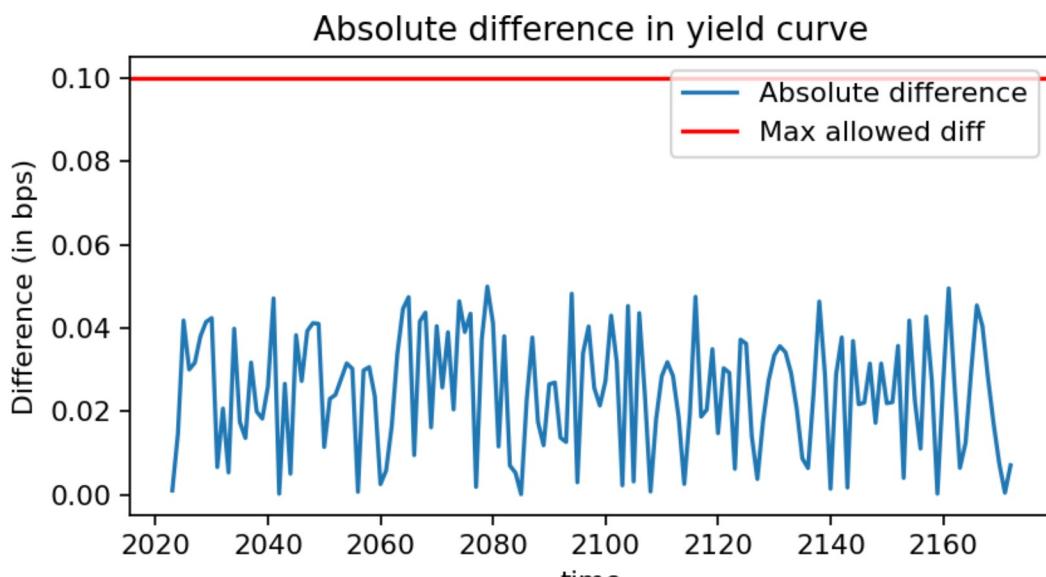
In [245...]

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

```
Test passed  
Test passed
```

In [246...]

```
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 April 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\_20230430\_Qb\_SW.xlsx* and the parameters displayed in the file *EIOPA\_RFR\_20230430\_Term\_Structures.xlsx*.

In [247...]

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

Out[247...]

	Mean test	Max test
Provided vs calculated	True	True

## Final yield curve

Full yield curve provided by EIOPA in %

In [248...]

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

Out[248...]

Country	
1	3.673
2	3.362
3	3.128
4	2.998
5	2.932
6	2.893
7	2.872
8	2.865
9	2.866
10	2.875
11	2.890
12	2.896
13	2.903
14	2.905
15	2.895
16	2.869
17	2.834
18	2.798
19	2.764
20	2.738
21	2.720
22	2.710
23	2.705
24	2.705
25	2.709
26	2.715
27	2.723
28	2.732

29	2.743
30	2.754
31	2.766
32	2.779
33	2.791
34	2.804
35	2.817
36	2.829
37	2.842
38	2.854
39	2.866
40	2.878
41	2.889
42	2.901
43	2.911
44	2.922
45	2.932
46	2.942
47	2.952
48	2.962
49	2.971
50	2.980
51	2.988
52	2.997
53	3.005
54	3.012
55	3.020
56	3.027
57	3.035
58	3.041
59	3.048
60	3.055
61	3.061
62	3.067
63	3.073
64	3.079
65	3.084
66	3.090
67	3.095
68	3.100
69	3.105
70	3.110
71	3.115
72	3.119
73	3.124
74	3.128
75	3.133
76	3.137
77	3.141
78	3.145
79	3.149
80	3.152
81	3.156
82	3.160
83	3.163
84	3.166
85	3.170
86	3.173
87	3.176

88	3.179
89	3.182
90	3.185
91	3.188
92	3.191
93	3.194
94	3.197
95	3.199
96	3.202
97	3.204
98	3.207
99	3.209
100	3.212
101	3.214
102	3.216
103	3.219
104	3.221
105	3.223
106	3.225
107	3.227
108	3.229
109	3.231
110	3.233
111	3.235
112	3.237
113	3.239
114	3.241
115	3.243
116	3.245
117	3.246
118	3.248
119	3.250
120	3.251
121	3.253
122	3.255
123	3.256
124	3.258
125	3.259
126	3.261
127	3.262
128	3.264
129	3.265
130	3.267
131	3.268
132	3.269
133	3.271
134	3.272
135	3.273
136	3.275
137	3.276
138	3.277
139	3.279
140	3.280
141	3.281
142	3.282
143	3.283
144	3.284
145	3.286
146	3.287

147 3.288  
148 3.289  
149 3.290  
150 3.291  
151 3.292  
152 3.293  
153 3.294  
154 3.295  
155 3.296  
156 3.297  
157 3.298  
158 3.299  
159 3.300  
160 3.301  
161 3.302  
162 3.303  
163 3.304  
164 3.305  
165 3.306  
166 3.307  
167 3.308  
168 3.309  
169 3.310  
170 3.311  
171 3.312  
172 3.313  
173 3.314  
174 3.315  
175 3.316  
176 3.317  
177 3.318  
178 3.319  
179 3.320  
180 3.321  
181 3.322  
182 3.323  
183 3.324  
184 3.325  
185 3.326  
186 3.327  
187 3.328  
188 3.329  
189 3.330  
190 3.331  
191 3.332  
192 3.333  
193 3.334  
194 3.335  
195 3.336  
196 3.337  
197 3.338  
198 3.339  
199 3.340  
200 3.341  
201 3.342  
202 3.343  
203 3.344  
204 3.345  
205 3.346  
206 3.347  
207 3.348  
208 3.349  
209 3.350  
210 3.351  
211 3.352  
212 3.353  
213 3.354  
214 3.355  
215 3.356  
216 3.357  
217 3.358  
218 3.359  
219 3.360  
220 3.361  
221 3.362  
222 3.363  
223 3.364  
224 3.365  
225 3.366  
226 3.367  
227 3.368  
228 3.369  
229 3.370  
230 3.371  
231 3.372  
232 3.373  
233 3.374  
234 3.375  
235 3.376  
236 3.377  
237 3.378  
238 3.379  
239 3.380  
240 3.381  
241 3.382  
242 3.383  
243 3.384  
244 3.385  
245 3.386  
246 3.387  
247 3.388  
248 3.389  
249 3.390  
250 3.391  
251 3.392  
252 3.393  
253 3.394  
254 3.395  
255 3.396  
256 3.397  
257 3.398  
258 3.399  
259 3.400  
260 3.401  
261 3.402  
262 3.403  
263 3.404  
264 3.405  
265 3.406  
266 3.407  
267 3.408  
268 3.409  
269 3.410  
270 3.411  
271 3.412  
272 3.413  
273 3.414  
274 3.415  
275 3.416  
276 3.417  
277 3.418  
278 3.419  
279 3.420  
280 3.421  
281 3.422  
282 3.423  
283 3.424  
284 3.425  
285 3.426  
286 3.427  
287 3.428  
288 3.429  
289 3.430  
290 3.431  
291 3.432  
292 3.433  
293 3.434  
294 3.435  
295 3.436  
296 3.437  
297 3.438  
298 3.439  
299 3.440  
300 3.441  
301 3.442  
302 3.443  
303 3.444  
304 3.445  
305 3.446  
306 3.447  
307 3.448  
308 3.449  
309 3.450  
310 3.451  
311 3.452  
312 3.453  
313 3.454  
314 3.455  
315 3.456  
316 3.457  
317 3.458  
318 3.459  
319 3.460  
320 3.461  
321 3.462  
322 3.463  
323 3.464  
324 3.465  
325 3.466  
326 3.467  
327 3.468  
328 3.469  
329 3.470  
330 3.471  
331 3.472  
332 3.473  
333 3.474  
334 3.475  
335 3.476  
336 3.477  
337 3.478  
338 3.479  
339 3.480  
340 3.481  
341 3.482  
342 3.483  
343 3.484  
344 3.485  
345 3.486  
346 3.487  
347 3.488  
348 3.489  
349 3.490  
350 3.491  
351 3.492  
352 3.493  
353 3.494  
354 3.495  
355 3.496  
356 3.497  
357 3.498  
358 3.499  
359 3.500  
360 3.501  
361 3.502  
362 3.503  
363 3.504  
364 3.505  
365 3.506  
366 3.507  
367 3.508  
368 3.509  
369 3.510  
370 3.511  
371 3.512  
372 3.513  
373 3.514  
374 3.515  
375 3.516  
376 3.517  
377 3.518  
378 3.519  
379 3.520  
380 3.521  
381 3.522  
382 3.523  
383 3.524  
384 3.525  
385 3.526  
386 3.527  
387 3.528  
388 3.529  
389 3.530  
390 3.531  
391 3.532  
392 3.533  
393 3.534  
394 3.535  
395 3.536  
396 3.537  
397 3.538  
398 3.539  
399 3.540  
400 3.541  
401 3.542  
402 3.543  
403 3.544  
404 3.545  
405 3.546  
406 3.547  
407 3.548  
408 3.549  
409 3.550  
410 3.551  
411 3.552  
412 3.553  
413 3.554  
414 3.555  
415 3.556  
416 3.557  
417 3.558  
418 3.559  
419 3.560  
420 3.561  
421 3.562  
422 3.563  
423 3.564  
424 3.565  
425 3.566  
426 3.567  
427 3.568  
428 3.569  
429 3.570  
430 3.571  
431 3.572  
432 3.573  
433 3.574  
434 3.575  
435 3.576  
436 3.577  
437 3.578  
438 3.579  
439 3.580  
440 3.581  
441 3.582  
442 3.583  
443 3.584  
444 3.585  
445 3.586  
446 3.587  
447 3.588  
448 3.589  
449 3.590  
450 3.591  
451 3.592  
452 3.593  
453 3.594  
454 3.595  
455 3.596  
456 3.597  
457 3.598  
458 3.599  
459 3.600  
460 3.601  
461 3.602  
462 3.603  
463 3.604  
464 3.605  
465 3.606  
466 3.607  
467 3.608  
468 3.609  
469 3.610  
470 3.611  
471 3.612  
472 3.613  
473 3.614  
474 3.615  
475 3.616  
476 3.617  
477 3.618  
478 3.619  
479 3.620  
480 3.621  
481 3.622  
482 3.623  
483 3.624  
484 3.625  
485 3.626  
486 3.627  
487 3.628  
488 3.629  
489 3.630  
490 3.631  
491 3.632  
492 3.633  
493 3.634  
494 3.635  
495 3.636  
496 3.637  
497 3.638  
498 3.639  
499 3.640  
500 3.641  
501 3.642  
502 3.643  
503 3.644  
504 3.645  
505 3.646  
506 3.647  
507 3.648  
508 3.649  
509 3.650  
510 3.651  
511 3.652  
512 3.653  
513 3.654  
514 3.655  
515 3.656  
516 3.657  
517 3.658  
518 3.659  
519 3.660  
520 3.661  
521 3.662  
522 3.663  
523 3.664  
524 3.665  
525 3.666  
526 3.667  
527 3.668  
528 3.669  
529 3.670  
530 3.671  
531 3.672  
532 3.673  
533 3.674  
534 3.675  
535 3.676  
536 3.677  
537 3.678  
538 3.679  
539 3.680  
540 3.681  
541 3.682  
542 3.683  
543 3.684  
544 3.685  
545 3.686  
546 3.687  
547 3.688  
548 3.689  
549 3.690  
550 3.691  
551 3.692  
552 3.693  
553 3.694  
554 3.695  
555 3.696  
556 3.697  
557 3.698  
558 3.699  
559 3.700  
560 3.701  
561 3.702  
562 3.703  
563 3.704  
564 3.705  
565 3.706  
566 3.707  
567 3.708  
568 3.709  
569 3.710  
570 3.711  
571 3.712  
572 3.713  
573 3.714  
574 3.715  
575 3.716  
576 3.717  
577 3.718  
578 3.719  
579 3.720  
580 3.721  
581 3.722  
582 3.723  
583 3.724  
584 3.725  
585 3.726  
586 3.727  
587 3.728  
588 3.729  
589 3.730  
590 3.731  
591 3.732  
592 3.733  
593 3.734  
594 3.735  
595 3.736  
596 3.737  
597 3.738  
598 3.739  
599 3.740  
600 3.741  
601 3.742  
602 3.743  
603 3.744  
604 3.745  
605 3.746  
606 3.747  
607 3.748  
608 3.749  
609 3.750  
610 3.751  
611 3.752  
612 3.753  
613 3.754  
614 3.755  
615 3.756  
616 3.757  
617 3.758  
618 3.759  
619 3.760  
620 3.761  
621 3.762  
622 3.763  
623 3.764  
624 3.765  
625 3.766  
626 3.767  
627 3.768  
628 3.769  
629 3.770  
630 3.771  
631 3.772  
632 3.773  
633 3.774  
634 3.775  
635 3.776  
636 3.777  
637 3.778  
638 3.779  
639 3.780  
640 3.781  
641 3.782  
642 3.783  
643 3.784  
644 3.785  
645 3.786  
646 3.787  
647 3.788  
648 3.789  
649 3.790  
650 3.791  
651 3.792  
652 3.793  
653 3.794  
654 3.795  
655 3.796  
656 3.797  
657 3.798  
658 3.799  
659 3.800  
660 3.801  
661 3.802  
662 3.803  
663 3.804  
664 3.805  
665 3.806  
666 3.807  
667 3.808  
668 3.809  
669 3.810  
670 3.811  
671 3.812  
672 3.813  
673 3.814  
674 3.815  
675 3.816  
676 3.817  
677 3.818  
678 3.819  
679 3.820  
680 3.821  
681 3.822  
682 3.823  
683 3.824  
684 3.825  
685 3.826  
686 3.827  
687 3.828  
688 3.829  
689 3.830  
690 3.831  
691 3.832  
692 3.833  
693 3.834  
694 3.835  
695 3.836  
696 3.837  
697 3.838  
698 3.839  
699 3.840  
700 3.841  
701 3.842  
702 3.843  
703 3.844  
704 3.845  
705 3.846  
706 3.847  
707 3.848  
708 3.849  
709 3.850  
710 3.851  
711 3.852  
712 3.853  
713 3.854  
714 3.855  
715 3.856  
716 3.857  
717 3.858  
718 3.859  
719 3.860  
720 3.861  
721 3.862  
722 3.863  
723 3.864  
724 3.865  
725 3.866  
726 3.867  
727 3.868  
728 3.869  
729 3.870  
730 3.871  
731 3.872  
732 3.873  
733 3.874  
734 3.875  
735 3.876  
736 3.877  
737 3.878  
738 3.879  
739 3.880  
740 3.881  
741 3.882  
742 3.883  
743 3.884  
744 3.885  
745 3.886  
746 3.887  
747 3.888  
748 3.889  
749 3.890  
750 3.891  
751 3.892  
752 3.893  
753 3.894  
754 3.895  
755 3.896  
756 3.897  
757 3.898  
758 3.899  
759 3.900  
760 3.901  
761 3.902  
762 3.903  
763 3.904  
764 3.905  
765 3.906  
766 3.907  
767 3.908  
768 3.909  
769 3.910  
770 3.911  
771 3.912  
772 3.913  
773 3.914  
774 3.915  
775 3.916  
776 3.917  
777 3.918  
778 3.919  
779 3.920  
780 3.921  
781 3.922  
782 3.923  
783 3.924  
784 3.925  
785 3.926  
786 3.927  
787 3.928  
788 3.929  
789 3.930  
790 3.931  
791 3.932  
792 3.933  
793 3.934  
794 3.935  
795 3.936  
796 3.937  
797 3.938  
798 3.939  
799 3.940  
800 3.941  
801 3.942  
802 3.943  
803 3.944  
804 3.945  
805 3.946  
806 3.947  
807 3.948  
808 3.949  
809 3.950  
810 3.951  
811 3.952  
812 3.953  
813 3.954  
814 3.955  
815 3.956  
816 3.957  
817 3.958  
818 3.959  
819 3.960  
820 3.961  
821 3.962  
822 3.963  
823 3.964  
824 3.965  
825 3.966  
826 3.967  
827 3.968  
828 3.969  
829 3.970  
830 3.971  
831 3.972  
832 3.973  
833 3.974  
834 3.975  
835 3.976  
836 3.977  
837 3.978  
838 3.979  
839 3.980  
840 3.981  
841 3.982  
842 3.983  
843 3.984  
844 3.985  
845 3.986  
846 3.987  
847 3.988  
848 3.989  
849 3.990  
850 3.991  
851 3.992  
852 3.993  
853 3.994  
854 3.995  
855 3.996  
856 3.997  
857 3.998  
858 3.999  
859 4.000  
860 4.001  
861 4.002  
862 4.003  
863 4.004  
864 4.005  
865 4.006  
866 4.007  
867 4.008  
868 4.009  
869 4.010  
870 4.011  
871 4.012  
872 4.013  
873 4.014  
874 4.015  
875 4.016  
876 4.017  
877 4.018  
878 4.019  
879 4.020  
880 4.021  
881 4.022  
882 4.023  
883 4.024  
884 4.025  
885 4.026  
886 4.027  
887 4.028  
888 4.029  
889 4.030  
890 4.031  
891 4.032  
892 4.033  
893 4.034  
894 4.035  
895 4.036  
896 4.037  
897 4.038  
898 4.039  
899 4.040  
900 4.041  
901 4.042  
902 4.043  
903 4.044  
904 4.045  
905 4.046  
906 4.047  
907 4.048  
908 4.049  
909 4.050  
910 4.051  
911 4.052  
912 4.053  
913 4.054  
914 4.055  
915 4.056  
916 4.057  
917 4.058  
918 4.059  
919 4.060  
920 4.061  
921 4.062  
922 4.063  
923 4.064  
924 4.065  
925 4.066  
926 4.067  
927 4.068  
928 4.069  
929 4.070  
930 4.071  
931 4.072  
932 4.073  
933 4.074  
934 4.075  
935 4.076  
936 4.077  
937 4.078  
938 4.079  
939 4.080  
940 4.081  
941 4.082  
942 4.083  
943 4.084  
944 4.085  
945 4.086  
946 4.087  
947 4.088  
948 4.089  
949 4.090  
950 4.091  
951 4.092  
952 4.093  
953 4.094  
954 4.095  
955 4.096  
956 4.097  
957 4.098  
958 4.099  
959 4.100  
960 4.101  
961 4.102  
962 4.103  
963 4.104  
964 4.105  
965 4.106  
966 4.107  
967 4.108  
968 4.109  
969 4.110  
970 4.111  
971 4.112  
972 4.113  
973 4.114  
974 4.115  
975 4.116  
976 4.117  
977 4.118  
978 4.119  
979 4.120  
980 4.121  
981 4.122  
982 4.123  
983 4.124  
984 4.125  
985 4.126  
986 4.127  
987 4.128  
988 4.129  
989 4.130  
990 4.131  
991 4.132  
992 4.133  
993 4.134  
994 4.135  
995 4.136  
996 4.137  
997 4.138  
998 4.139  
999 4.140  
1000 4.141  
1001 4.142  
1002 4.143  
1003 4.144  
1004 4.145  
1005 4.146  
1006 4.147  
1007 4.148  
1008 4.149  
1009 4.150  
1010 4.151  
1011 4.152  
1012 4.153  
1013 4.154  
1014 4.155  
1015 4.156  
1016 4.157  
1017 4.158  
1018 4.159  
1019 4.160  
1020 4.161  
1021 4.162  
1022 4.163  
1023 4.164  
1024 4.165  
1025 4.166  
1026 4.167  
1027 4.168  
1028 4.169  
1029 4.170  
1030 4.171  
1031 4.172  
1032 4.173  
1033 4.174  
1034 4.175  
1035 4.176  
1036 4.177  
1037 4.178  
1038 4.179  
1039 4.180  
1040 4.181  
1041 4.182  
1042 4.183  
1043 4.184  
1044 4.185  
1045 4.186  
1046 4.187  
1047 4.188  
1048 4.189  
1049 4.190  
1050 4.191  
1051 4.192  
1052 4.193  
1053 4.194  
1054 4.195  
1055 4.196  
1056 4.197  
1057 4.198  
1058 4.199  
1059 4.200  
1060 4.201  
1061 4.202  
1062 4.203  
1063 4.204  
1064 4.205  
1065 4.206  
1066 4.207  
1067 4.208  
1068 4.209  
1069 4.210  
1070 4.211  
1071 4.212  
1072 4.213  
1073 4.214  
1074 4.215  
1075 4.216  
1076 4.217  
1077 4.218  
1078 4.219  
1079 4.220  
1080 4.221  
1081 4.222  
1082 4.223  
1083 4.224  
1084 4.225  
1085 4.226  
1086 4.227  
1087 4.228  
1088 4.229  
1089 4.230  
1090 4.231  
1091 4.232  
1092 4.233  
1093 4.234  
1094 4.235  
1095 4.236  
1096 4.237  
1097 4.238  
1098 4.239  
1099 4.240  
1100 4.241  
1101 4.242  
1102 4.243  
1103 4.244  
1104 4.245  
1105 4.246  
1106 4.247  
1107 4.248  
1108 4.249  
1109 4.250  
1110 4.251  
1111 4.252  
1112 4.253  
1113 4.254  
1114 4.255  
1115 4.256  
1116 4.257  
1117 4.258  
1118 4.259  
1119 4.260  
1120 4.261  
1121 4.262  
1122 4.263  
1123 4.264  
1124 4.265  
1125 4.266  
1126 4.267  
1127 4.268  
1128 4.269  
1129 4.270  
1130 4.271  
1131 4.272  
1132 4.273  
1133 4.274  
1134 4.275  
1135 4.276  
1136 4.277  
1137 4.278  
1138 4.279  
1139 4.280  
1140 4.281  
1141 4.282  
1142 4.283  
1143 4.284  
1144 4.285  
1145 4.286  
1146 4.287  
1147 4.288  
1148 4.289  
1149 4.290  
11