

# 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 [188...]

```
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 [189...]

```
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 [190...]

```
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 [191...]

```
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 [192...]

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

## Parameter input

Parameters sheet

In [193...]

```
param_raw.head()
```

Out[193...]

	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>	0.111987	0.111987	0.111987	0.111987	0.111987	0.111987

5 rows × 106 columns

The country selected is:

In [194...]

```
country = "Italy"
```

In [195...]

```
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 [196...]

```
extra_param
```

Out[196...]

Country	
Coupon_freq	1.000000
LLP	20.000000
Convergence	40.000000
UFR	3.450000
alpha	0.111987
CRA	10.000000

Name: Italy\_Values, dtype: float64

In [197...]

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

In [198...]

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

## Maturity vector

Vector of maturities used in the calibration

In [199...]

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

Out[199...]

```
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: Italy_Maturities, dtype: float64
```

In [200...]

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

## Calibration vector

Vector **Qb** provided as input

In [201...]

```
Qb
```

Out[201...]

```
Country
1      -1.454577
2      -4.475501
3      1.142665
4      0.583288
5      1.482629
6      -1.144625
7      0.883939
8      -0.463093
9      0.129968
10     1.076823
11     -1.775038
12     1.508691
13     -0.024800
14     -0.023458
15     -1.378730
16     0.020643
```

```
17    0.054165
18   -0.081945
19    0.420393
20    0.329203
Name: Italy Values, dtype: float64
```

```
In [202...]: curve_raw = pd.read_csv(selected_curves_file, sep=',', index_col=0)
```

```
In [203...]: curve_country = curve_raw.loc[:, country]
```

[Back to the top](#)

## Calibration parameters and calibration vector provided by EIOPA

```
In [204...]: # 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 [205...]

```

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 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 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
    #
    # 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 [206...]

```

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 [207...]

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

Out[207...]

#### Recalculated rates

0	0.041930
1	0.039821
2	0.037113
3	0.034921
4	0.033423
5	0.032465
6	0.031797
7	0.031360
8	0.031069
9	0.030895
10	0.030795
11	0.030690
12	0.030606
13	0.030493
14	0.030291

---

[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 [208...]

```
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 [209...]

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

### EIOPA curve provided

Yield curve provided by EIOPA

In [210...]

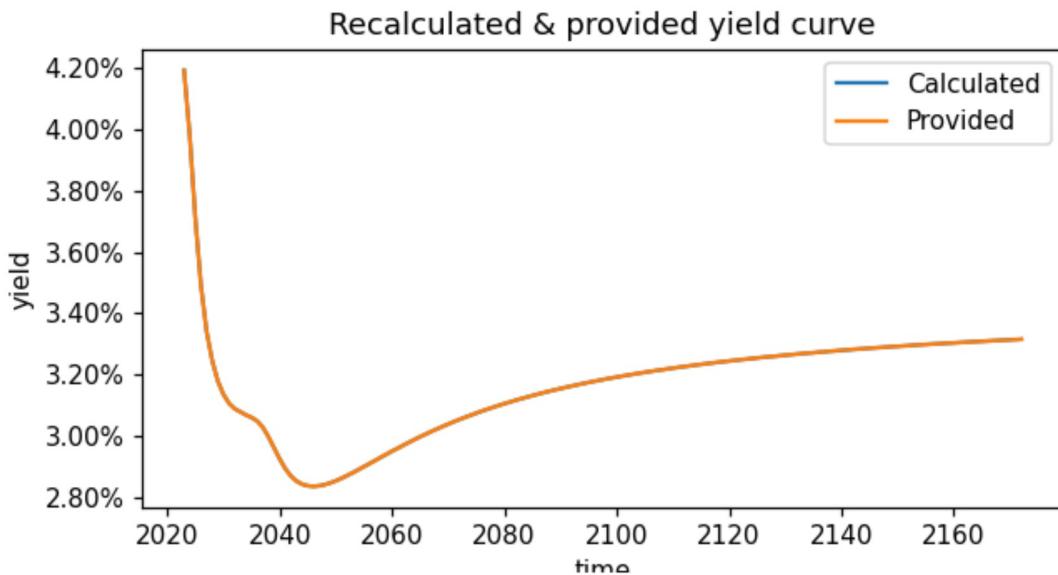
```
target_curve.head()
```

Out[210...]

	Given rates
<b>0</b>	0.04193
<b>1</b>	0.03982
<b>2</b>	0.03711
<b>3</b>	0.03492
<b>4</b>	0.03342

```
In [211...]:  
x_data_label = range(2023,2023+r_Target.shape[0],1)
```

```
In [212...]:  
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 [213...]:  
test_statistics_bdp = pd.DataFrame(abs(r_Target.values-target_curve.values)*10000, co
```

### EIOPA curve comparison

Absolute difference in bps

```
In [214...]:  
test_statistics_bdp.head()
```

```
Out[214...]:  
Abs diff in bps
```

0	6.187075e-07
1	9.984608e-03
2	2.859527e-02
3	1.251384e-02

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.

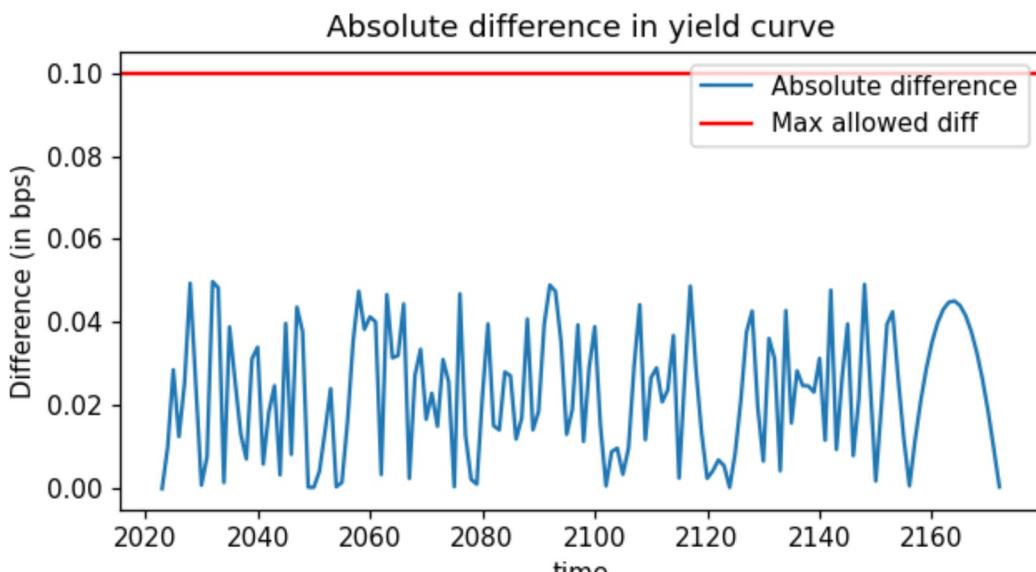
In [215...]

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

```
Test passed  
Test passed
```

In [216...]

```
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 [217...]

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

Out[217...]

	Mean test	Max test
Provided vs calculated	True	True

### Final yield curve

Full yield curve provided by EIOPA in %

In [218...]

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

Out[218...]

Country	
1	4.193
2	3.982
3	3.711
4	3.492
5	3.342
6	3.247
7	3.180
8	3.136
9	3.107
10	3.089
11	3.079
12	3.069
13	3.061
14	3.049
15	3.029
16	2.998
17	2.962
18	2.926
19	2.894
20	2.870
21	2.853
22	2.843
23	2.838
24	2.836
25	2.838
26	2.841
27	2.847
28	2.854

29	2.862
30	2.871
31	2.880
32	2.890
33	2.900
34	2.910
35	2.920
36	2.931
37	2.941
38	2.951
39	2.960
40	2.970
41	2.979
42	2.989
43	2.998
44	3.006
45	3.015
46	3.023
47	3.031
48	3.039
49	3.047
50	3.054
51	3.061
52	3.068
53	3.075
54	3.082
55	3.088
56	3.094
57	3.100
58	3.106
59	3.111
60	3.117
61	3.122
62	3.127
63	3.132
64	3.137
65	3.142
66	3.146
67	3.151
68	3.155
69	3.159
70	3.163
71	3.167
72	3.171
73	3.175
74	3.179
75	3.182
76	3.186
77	3.189
78	3.193
79	3.196
80	3.199
81	3.202
82	3.205
83	3.208
84	3.211
85	3.214
86	3.216
87	3.219

88	3.222
89	3.224
90	3.227
91	3.229
92	3.232
93	3.234
94	3.236
95	3.239
96	3.241
97	3.243
98	3.245
99	3.247
100	3.249
101	3.251
102	3.253
103	3.255
104	3.257
105	3.259
106	3.260
107	3.262
108	3.264
109	3.266
110	3.267
111	3.269
112	3.271
113	3.272
114	3.274
115	3.275
116	3.277
117	3.278
118	3.280
119	3.281
120	3.283
121	3.284
122	3.285
123	3.287
124	3.288
125	3.289
126	3.290
127	3.292
128	3.293
129	3.294
130	3.295
131	3.297
132	3.298
133	3.299
134	3.300
135	3.301
136	3.302
137	3.303
138	3.304
139	3.305
140	3.306
141	3.307
142	3.308
143	3.309
144	3.310
145	3.311
146	3.312

147 3.313  
148 3.314  
149 3.315  
150 3.316  
151 3.317  
152 3.318  
153 3.319  
154 3.320  
155 3.321  
156 3.322  
157 3.323  
158 3.324  
159 3.325  
160 3.326  
161 3.327  
162 3.328  
163 3.329  
164 3.330  
165 3.331  
166 3.332  
167 3.333  
168 3.334  
169 3.335  
170 3.336  
171 3.337  
172 3.338  
173 3.339  
174 3.340  
175 3.341  
176 3.342  
177 3.343  
178 3.344  
179 3.345  
180 3.346  
181 3.347  
182 3.348  
183 3.349  
184 3.350  
185 3.351  
186 3.352  
187 3.353  
188 3.354  
189 3.355  
190 3.356  
191 3.357  
192 3.358  
193 3.359  
194 3.360  
195 3.361  
196 3.362  
197 3.363  
198 3.364  
199 3.365  
200 3.366  
201 3.367  
202 3.368  
203 3.369  
204 3.370  
205 3.371  
206 3.372  
207 3.373  
208 3.374  
209 3.375  
210 3.376  
211 3.377  
212 3.378  
213 3.379  
214 3.380  
215 3.381  
216 3.382  
217 3.383  
218 3.384  
219 3.385  
220 3.386  
221 3.387  
222 3.388  
223 3.389  
224 3.390  
225 3.391  
226 3.392  
227 3.393  
228 3.394  
229 3.395  
230 3.396  
231 3.397  
232 3.398  
233 3.399  
234 3.400  
235 3.401  
236 3.402  
237 3.403  
238 3.404  
239 3.405  
240 3.406  
241 3.407  
242 3.408  
243 3.409  
244 3.410  
245 3.411  
246 3.412  
247 3.413  
248 3.414  
249 3.415  
250 3.416  
251 3.417  
252 3.418  
253 3.419  
254 3.420  
255 3.421  
256 3.422  
257 3.423  
258 3.424  
259 3.425  
260 3.426  
261 3.427  
262 3.428  
263 3.429  
264 3.430  
265 3.431  
266 3.432  
267 3.433  
268 3.434  
269 3.435  
270 3.436  
271 3.437  
272 3.438  
273 3.439  
274 3.440  
275 3.441  
276 3.442  
277 3.443  
278 3.444  
279 3.445  
280 3.446  
281 3.447  
282 3.448  
283 3.449  
284 3.450  
285 3.451  
286 3.452  
287 3.453  
288 3.454  
289 3.455  
290 3.456  
291 3.457  
292 3.458  
293 3.459  
294 3.460  
295 3.461  
296 3.462  
297 3.463  
298 3.464  
299 3.465  
300 3.466  
301 3.467  
302 3.468  
303 3.469  
304 3.470  
305 3.471  
306 3.472  
307 3.473  
308 3.474  
309 3.475  
310 3.476  
311 3.477  
312 3.478  
313 3.479  
314 3.480  
315 3.481  
316 3.482  
317 3.483  
318 3.484  
319 3.485  
320 3.486  
321 3.487  
322 3.488  
323 3.489  
324 3.490  
325 3.491  
326 3.492  
327 3.493  
328 3.494  
329 3.495  
330 3.496  
331 3.497  
332 3.498  
333 3.499  
334 3.500  
335 3.501  
336 3.502  
337 3.503  
338 3.504  
339 3.505  
340 3.506  
341 3.507  
342 3.508  
343 3.509  
344 3.510  
345 3.511  
346 3.512  
347 3.513  
348 3.514  
349 3.515  
350 3.516  
351 3.517  
352 3.518  
353 3.519  
354 3.520  
355 3.521  
356 3.522  
357 3.523  
358 3.524  
359 3.525  
360 3.526  
361 3.527  
362 3.528  
363 3.529  
364 3.530  
365 3.531  
366 3.532  
367 3.533  
368 3.534  
369 3.535  
370 3.536  
371 3.537  
372 3.538  
373 3.539  
374 3.540  
375 3.541  
376 3.542  
377 3.543  
378 3.544  
379 3.545  
380 3.546  
381 3.547  
382 3.548  
383 3.549  
384 3.550  
385 3.551  
386 3.552  
387 3.553  
388 3.554  
389 3.555  
390 3.556  
391 3.557  
392 3.558  
393 3.559  
394 3.560  
395 3.561  
396 3.562  
397 3.563  
398 3.564  
399 3.565  
400 3.566  
401 3.567  
402 3.568  
403 3.569  
404 3.570  
405 3.571  
406 3.572  
407 3.573  
408 3.574  
409 3.575  
410 3.576  
411 3.577  
412 3.578  
413 3.579  
414 3.580  
415 3.581  
416 3.582  
417 3.583  
418 3.584  
419 3.585  
420 3.586  
421 3.587  
422 3.588  
423 3.589  
424 3.590  
425 3.591  
426 3.592  
427 3.593  
428 3.594  
429 3.595  
430 3.596  
431 3.597  
432 3.598  
433 3.599  
434 3.600  
435 3.601  
436 3.602  
437 3.603  
438 3.604  
439 3.605  
440 3.606  
441 3.607  
442 3.608  
443 3.609  
444 3.610  
445 3.611  
446 3.612  
447 3.613  
448 3.614  
449 3.615  
450 3.616  
451 3.617  
452 3.618  
453 3.619  
454 3.620  
455 3.621  
456 3.622  
457 3.623  
458 3.624  
459 3.625  
460 3.626  
461 3.627  
462 3.628  
463 3.629  
464 3.630  
465 3.631  
466 3.632  
467 3.633  
468 3.634  
469 3.635  
470 3.636  
471 3.637  
472 3.638  
473 3.639  
474 3.640  
475 3.641  
476 3.642  
477 3.643  
478 3.644  
479 3.645  
480 3.646  
481 3.647  
482 3.648  
483 3.649  
484 3.650  
485 3.651  
486 3.652  
487 3.653  
488 3.654  
489 3.655  
490 3.656  
491 3.657  
492 3.658  
493 3.659  
494 3.660  
495 3.661  
496 3.662  
497 3.663  
498 3.664  
499 3.665  
500 3.666  
501 3.667  
502 3.668  
503 3.669  
504 3.670  
505 3.671  
506 3.672  
507 3.673  
508 3.674  
509 3.675  
510 3.676  
511 3.677  
512 3.678  
513 3.679  
514 3.680  
515 3.681  
516 3.682  
517 3.683  
518 3.684  
519 3.685  
520 3.686  
521 3.687  
522 3.688  
523 3.689  
524 3.690  
525 3.691  
526 3.692  
527 3.693  
528 3.694  
529 3.695  
530 3.696  
531 3.697  
532 3.698  
533 3.699  
534 3.700  
535 3.701  
536 3.702  
537 3.703  
538 3.704  
539 3.705  
540 3.706  
541 3.707  
542 3.708  
543 3.709  
544 3.710  
545 3.711  
546 3.712  
547 3.713  
548 3.714  
549 3.715  
550 3.716  
551 3.717  
552 3.718  
553 3.719  
554 3.720  
555 3.721  
556 3.722  
557 3.723  
558 3.724  
559 3.725  
560 3.726  
561 3.727  
562 3.728  
563 3.729  
564 3.730  
565 3.731  
566 3.732  
567 3.733  
568 3.734  
569 3.735  
570 3.736  
571 3.737  
572 3.738  
573 3.739  
574 3.740  
575 3.741  
576 3.742  
577 3.743  
578 3.744  
579 3.745  
580 3.746  
581 3.747  
582 3.748  
583 3.749  
584 3.750  
585 3.751  
586 3.752  
587 3.753  
588 3.754  
589 3.755  
590 3.756  
591 3.757  
592 3.758  
593 3.759  
594 3.760  
595 3.761  
596 3.762  
597 3.763  
598 3.764  
599 3.765  
600 3.766  
601 3.767  
602 3.768  
603 3.769  
604 3.770  
605 3.771  
606 3.772  
607 3.773  
608 3.774  
609 3.775  
610 3.776  
611 3.777  
612 3.778  
613 3.779  
614 3.780  
615 3.781  
616 3.782  
617 3.783  
618 3.784  
619 3.785  
620 3.786  
621 3.787  
622 3.788  
623 3.789  
624 3.790  
625 3.791  
626 3.792  
627 3.793  
628 3.794  
629 3.795  
630 3.796  
631 3.797  
632 3.798  
633 3.799  
634 3.800  
635 3.801  
636 3.802  
637 3.803  
638 3.804  
639 3.805  
640 3.806  
641 3.807  
642 3.808  
643 3.809  
644 3.810  
645 3.811  
646 3.812  
647 3.813  
648 3.814  
649 3.815  
650 3.816  
651 3.817  
652 3.818  
653 3.819  
654 3.820  
655 3.821  
656 3.822  
657 3.823  
658 3.824  
659 3.825  
660 3.826  
661 3.827  
662 3.828  
663 3.829  
664 3.830  
665 3.831  
666 3.832  
667 3.833  
668 3.834  
669 3.835  
670 3.836  
671 3.837  
672 3.838  
673 3.839  
674 3.840  
675 3.841  
676 3.842  
677 3.843  
678 3.844  
679 3.845  
680 3.846  
681 3.847  
682 3.848  
683 3.849  
684 3.850  
685 3.851  
686 3.852  
687 3.853  
688 3.854  
689 3.855  
690 3.856  
691 3.857  
692 3.858  
693 3.859  
694 3.860  
695 3.861  
696 3.862  
697 3.863  
698 3.864  
699 3.865  
700 3.866  
701 3.867  
702 3.868  
703 3.869  
704 3.870  
705 3.871  
706 3.872  
707 3.873  
708 3.874  
709 3.875  
710 3.876  
711 3.877  
712 3.878  
713 3.879  
714 3.880  
715 3.881  
716 3.882  
717 3.883  
718 3.884  
719 3.885  
720 3.886  
721 3.887  
722 3.888  
723 3.889  
724 3.890  
725 3.891  
726 3.892  
727 3.893  
728 3.894  
729 3.895  
730 3.896  
731 3.897  
732 3.898  
733 3.899  
734 3.900  
735 3.901  
736 3.902  
737 3.903  
738 3.904  
739 3.905  
740 3.906  
741 3.907  
742 3.908  
743 3.909  
744 3.910  
745 3.911  
746 3.912  
747 3.913  
748 3.914  
749 3.915  
750 3.916  
751 3.917  
752 3.918  
753 3.919  
754 3.920  
755 3.921  
756 3.922  
757 3.923  
758 3.924  
759 3.925  
760 3.926  
761 3.927  
762 3.928  
763 3.929  
764 3.930  
765 3.931  
766 3.932  
767 3.933  
768 3.934  
769 3.935  
770 3.936  
771 3.937  
772 3.938  
773 3.939  
774 3.940  
775 3.941  
776 3.942  
777 3.943  
778 3.944  
779 3.945  
780 3.946  
781 3.947  
782 3.948  
783 3.949  
784 3.950  
785 3.951  
786 3.952  
787 3.953  
788 3.954  
789 3.955  
790 3.956  
791 3.957  
792 3.958  
793 3.959  
794 3.960  
795 3.961  
796 3.962  
797 3.963  
798 3.964  
799 3.965  
800 3.966  
801 3.967  
802 3.968  
803 3.969  
804 3.970  
805 3.971  
806 3.972  
807 3.973  
808 3.974  
809 3.975  
810 3.976  
811 3.977  
812 3.978  
813 3.979  
814 3.980  
815 3.981  
816 3.982  
817 3.983  
818 3.984  
819 3.985  
820 3.986  
821 3.987  
822 3.988  
823 3.989  
824 3.990  
825 3.991  
826 3.992  
827 3.993  
828 3.994  
829 3.995  
830 3.996  
831 3.997  
832 3.998  
833 3.999  
834 4.000