

EIOPA RISK-FREE CURVE JULY-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 July 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_20230731_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_20230731_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	0.108242	0.108242	0.108242	0.108242	0.108242	0.108242

5 rows × 106 columns

The country selected is:

In [131...]

```
country = "Slovenia"
```

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              20.000000
Convergence      40.000000
UFR              3.450000
alpha             0.108242
CRA               10.000000
Name: Slovenia_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: Slovenia_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      -9.123120
2       0.761538
3       0.746101
4       0.811666
5       1.499239
6      -2.539173
7       2.742712
8      -2.558252
9       5.138513
10     -10.311679
11      12.034373
12     -5.290139
13      0.000036
14     -0.005738
15     -0.822778
16      0.018952
```

```
17      0.045746
18     -0.061888
19      0.334426
20      0.341258
Name: Slovenia 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 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 [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. 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 [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 **Qb**

In [144...]

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

Out[144...]

Recalculated rates

0	0.040790
1	0.037705
2	0.035172
3	0.033476
4	0.032460
5	0.031869
6	0.031463
7	0.031279
8	0.031201
9	0.031208
10	0.031143
11	0.031321
12	0.031437
13	0.031411
14	0.031256

[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_20230731_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...]

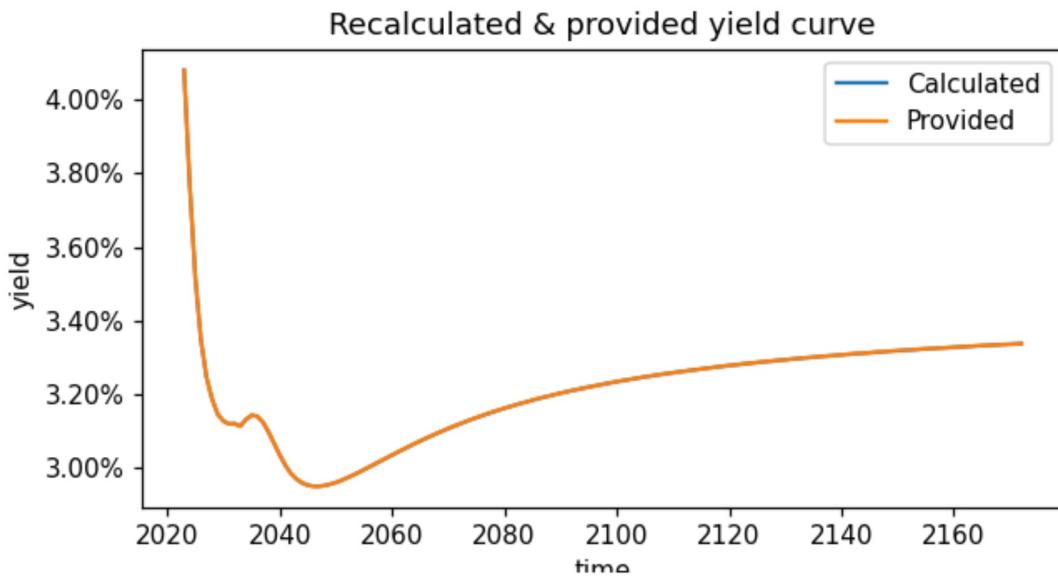
```
target_curve.head()
```

Out[147...]

	Given rates
0	0.04079
1	0.03771
2	0.03517
3	0.03348
4	0.03246

```
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
```

	Abs diff in bps
0	0.000001
1	0.046185
2	0.024588
3	0.040504

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.

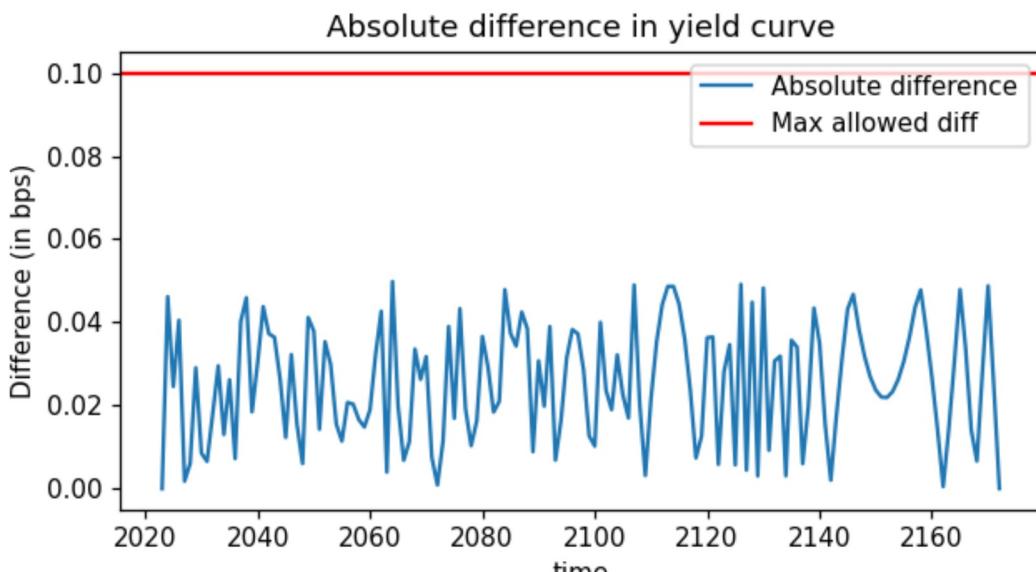
In [152...]

```
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 July 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_20230731_Qb_SW.xlsx* and the parameters displayed in the file *EIOPA_RFR_20230731_Term_Structures.xlsx*.

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	4.079
2	3.771
3	3.517
4	3.348
5	3.246
6	3.187
7	3.146
8	3.128
9	3.120
10	3.121
11	3.114
12	3.132
13	3.144
14	3.141
15	3.126
16	3.099
17	3.068
18	3.037
19	3.009
20	2.986
21	2.971
22	2.960
23	2.954
24	2.951
25	2.951
26	2.953
27	2.957
28	2.961

29	2.967
30	2.974
31	2.981
32	2.988
33	2.996
34	3.004
35	3.012
36	3.020
37	3.028
38	3.036
39	3.044
40	3.051
41	3.059
42	3.067
43	3.074
44	3.081
45	3.088
46	3.095
47	3.101
48	3.108
49	3.114
50	3.120
51	3.126
52	3.132
53	3.137
54	3.143
55	3.148
56	3.153
57	3.158
58	3.163
59	3.167
60	3.172
61	3.176
62	3.180
63	3.185
64	3.189
65	3.193
66	3.196
67	3.200
68	3.204
69	3.207
70	3.211
71	3.214
72	3.217
73	3.220
74	3.223
75	3.226
76	3.229
77	3.232
78	3.235
79	3.238
80	3.240
81	3.243
82	3.245
83	3.248
84	3.250
85	3.253
86	3.255
87	3.257

88	3.259
89	3.261
90	3.263
91	3.265
92	3.267
93	3.269
94	3.271
95	3.273
96	3.275
97	3.277
98	3.279
99	3.280
100	3.282
101	3.284
102	3.285
103	3.287
104	3.289
105	3.290
106	3.292
107	3.293
108	3.294
109	3.296
110	3.297
111	3.299
112	3.300
113	3.301
114	3.303
115	3.304
116	3.305
117	3.306
118	3.308
119	3.309
120	3.310
121	3.311
122	3.312
123	3.313
124	3.315
125	3.316
126	3.317
127	3.318
128	3.319
129	3.320
130	3.321
131	3.322
132	3.323
133	3.324
134	3.325
135	3.326
136	3.326
137	3.327
138	3.328
139	3.329
140	3.330
141	3.331
142	3.332
143	3.333
144	3.333
145	3.334
146	3.335

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